A Hardware Based 3D Room Scanner

R. P. Ramsay

A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Engineering
in
Electrical and Electronic Engineering
at the
University of Canterbury,
Christchurch, New Zealand.

31 March 2008

ABSTRACT

This thesis describes a project to create a hardware based 3D interior scanner. This was based on a previous project that created a scanner optimised for interior conditions, using structured light triangulation. The original project referred to as the Mark-I scanner, performed its control and processing on a PC and the primary goal of this project was to re-implement this system using hardware, making the scanner more portable and simpler to use. The Mark-I system required a specialised camera which had an unusually high noise associated with it, so a secondary goal was to investigate whether this camera could be replaced with a superior model or this noise corrected.

A Mark-II scanner system was created using FPGA processing and control implemented in the VHDL language. This read from a CMOS camera, controlled the system's motor and laser, generated 3D points and communicated with users. A suitable camera was not found and the Mark-I scanners camera was found to have been damaged and become unusable, so a simulation environment was constructed that simulated the operation of the scanner, created 3D images for it to process, and tested its results.

Chapter 1 of this thesis outlines the goals of this project and describes the Mark-I system. Chapter 2 describes the theory and properties of the Mark-I system, and chapter 3 describes the work undertaken to replace the scanner's sensor. Chapter 4 describes the system created to interface to CMOS sensors, and chapter 5 outlines the theory involved in calculating 3D points using structured light triangulation. The final hardware scanner, and the simulation system used to test it, are then described in chapter 6.

ACKNOWLEDGEMENTS

I would like to express deep gratitude to both my Parents, for encouraging and supporting my education throughout the years. This belongs to you.

Thanks goes to my supervisor Dr Andrew Bainbridge-Smith for his advice and support, and to Dr Michael Hayes and Timothy Llewellynn for their time. I would also like to thank Steve Weddell and Dave van Leeuwen for their invaluable assistance with my problems, and a big chur goes to Erwin and Raph for seeing me over the line.

CONTENTS

Chapter 1

INTRODUCTION

1.1 THE ORIGINAL PROJECT

The work outlined in this thesis is an extension of the work carried out for a previous master's thesis "Three Dimensional Interior Room Scanner" by T. Llewellynn [1]. Completed in 2001 this project had the aim of building a low cost 3D scanner for digitising indoor rooms.

1.1.1 Objectives

Then, and now, most three-dimensional imaging systems were designed for highly accurate very close range scanning of relatively small objects, particularly for digital content creation or medical applications [2]. Scanners with larger working ranges have been of interest in industrial automation and robotics, where large scanners, coarse resolutions, complex systems, or expensive solutions have been acceptable [3]. Because of this these systems are not suitable for consumer or mass production applications.

An interior 3D scanner would have many applications where accurate 3D images of rooms and buildings would be useful. Areas ranging from architecture, real-estate, and motion picture animation would all benefit from such a system. In the near future there may well be a large market for 3D scanners tailored to interior conditions, created by the Google Earth [4] and Windows Virtual Earth [5] projects. These plan to encourage companies and individuals to create 3D models of buildings, which may provide a market for mass market interior scanners. The lack of a fast, simple, and low cost solution for tasks of this nature was the motivation for this project.

The objective of the project was to create a scanning platform optimised for the lighting conditions and distances unique to interior rooms. This would be a small, portable system, producing high accuracy, medium range scans with a dense resolution map. The scanner was intended to be simple and practical to use, and aimed at a mass market by using common low cost components.

Its stated aims were:

- To be able to scan an indoor environment under normal indoor lighting conditions.
- Scan times are to be kept under an hour.
- An accuracy of approximately 1 part in 1000 over an operating range of 7 m is required.
- The scanner design should be rugged, so that positioning of the device is easy and not damaging to the device itself.
- The physical size of the device is to be no larger that 0.5 m in span in order to keep the scanner portable.
- It should be possible to perform multiple scans of a room interior, which are then combined into a single model.

The lighting requirements meant the scanner did not need to operate in direct sunlight, but did need to be tolerant to indoor lighting such as fluorescent bulbs. The limitation on scan times was intended to minimise disruption to busy environments.

1.1.2 Overview

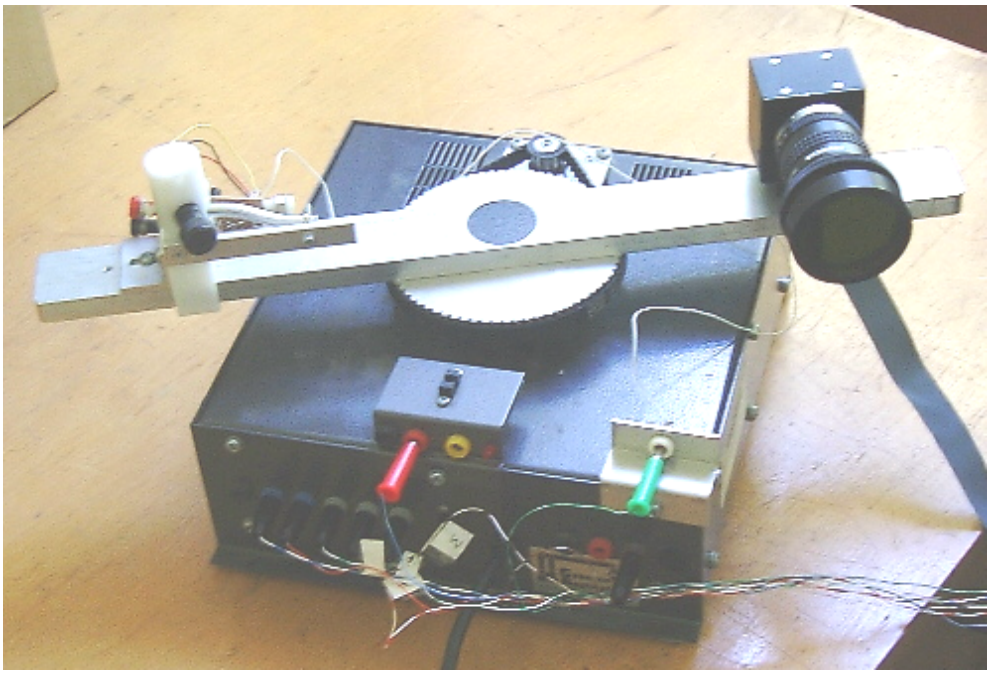


Figure 1.1 The Mark-I scanner created for the master's thesis [1].

The system developed to meet these goals is shown in figure 1.1, and will be referred to from here on as the Mark-I System. This used a method known as structured light triangulation to create the 3D points.

A laser is mounted on the right hand side of the scanner's arm. This projects a vertical plane of light onto the scene. This structured illumination reflects off the scene and is imaged by a CMOS camera on the left side of the scanner. This uses various filters to improve its detection of the laser line. The position of an illuminated sensor-pixel determines a straight ray along which the light that arrived at that pixel travelled. Trigonometry is then used to determine the point in 3D space that the reflection originated, by exploiting prior knowledge of the the scanner's geometry. This process shown in figure 1.2 is called "structured light triangulation", and is described in more detail in section 5.

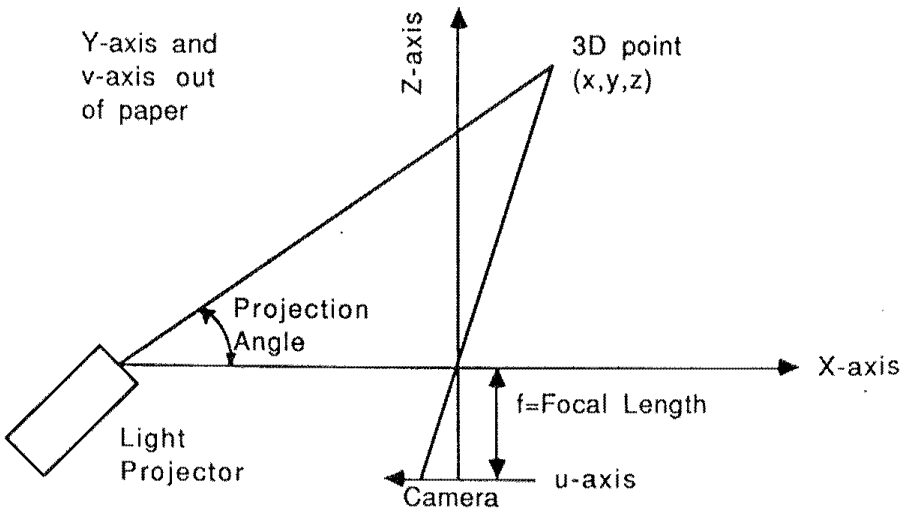


Figure 1.2 Structure light illumination, take from [6].

The arm is mounted on a stepper motor, because this can only image points along the laser's plane. The motor rotates the "rigid scanner" a set angle and a scan is taken. The motor then rotates another set amount and another scan is taken. In this way a full 360° scan of the room can be created. A side effect of this system is that the "cloud" of 3D point data it produces is organised into vertical stripes as a result of its planar light source. An example of a scan taken by this scanner is shown in figure 1.3.

A computer running custom designed software, on the Linux operating system, controlled the hardware and captured the resulting data. This computer controlled the motor, controlled the laser, read from the CMOS camera, and performed the calibration and point estimation algorithms for the system using the data received. The low level control was performed by software written in C, which accessed the computer's hardware. This included a custom made driver which allowed a Linux operating system to access the PCI card. These programs were controlled by Java software which performed the more complex tasks required. It controlled the scanner's operation, and calculated 3D points using the data received from the camera, allowed the 3D points collected to be viewed, and its interface is shown in figure 1.4.

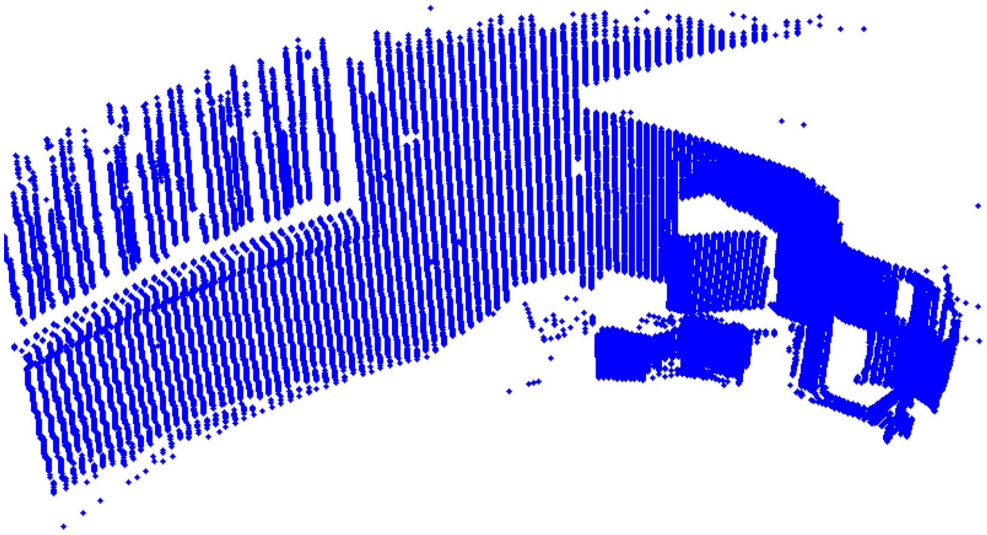


Figure 1.3 An image taken by the Mark-I scanner, taken from [1].

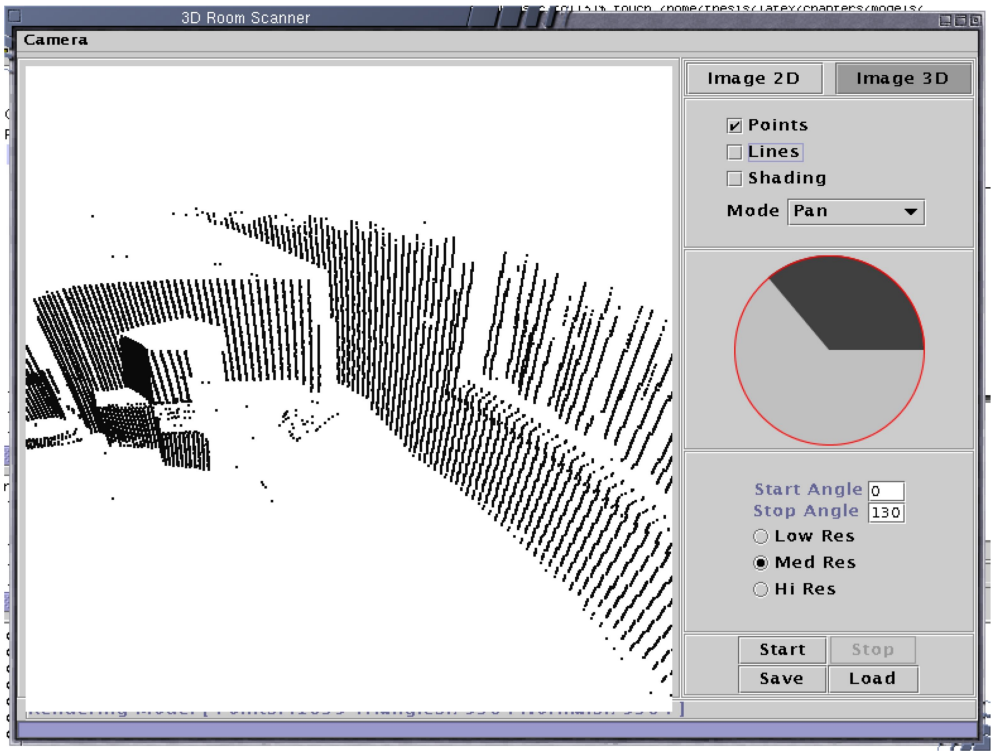


Figure 1.4 The system's software displaying a scan of an interior, taken from [1].

1.2 PROPERTIES

The wide distances and high accuracy requirements of indoor scanners meant that a wide dynamic range was required. Light reflecting off objects close to the scanner are brightly illuminated, while objects far away appear much darker. In order to image both kinds of objects the Mark-I scanner used a CMOS camera with a logarithmic response. This camera allowed the readout of individual pixels and regions, which can speed up scanning. However CMOS sensors have an inherent Fixed Pattern Noise (FPN) (see chapter 3), of random pixel offsets created during fabrication which had to be corrected using a nonlinear software correction of the points it produced [1].

The stripe produced by the laser spreads out rather than being a constant intensity [7]. This can be used to find the laser line's centre to an accuracy smaller than the width of the pixels used to image the laser. [1] showed that a Gaussian model produced very accurate centre detection, but the actual method used in the scanner simply finds the brightest pixel in each line by examining every pixel, while using a threshold to remove noise. The intent was to remove objects for which only noisy and inaccurate results could be computed, but it also meant that distant objects under the threshold were not found.

Calculating 3D points involved correcting for many physical processes. First lens distortions were removed. Pixel addresses must then be transformed into 2D data based on the sensor's dimensions, and 3D data must be generated from the 2D values using triangulation. These points may then be moved so they are expressed relative to the centre of the scanner, and then the rotation caused by the movement of the scanner removed. First the system removes lens distortion from the pixel data, then it performs the 3D triangulation, and then the rotations and translations are performed. The parameters for these two calculations are determined in a calibration process involving the scanning of multiple printed checkerboards. In this way the scanner's dimensions do not need to be known, and errors created in measuring these can be avoided.

1.2.1 Results

After its implementation the first system was evaluated against its specifications. Only simple centroiding was implemented but simulations showed that the advanced methods of locating the laser stripe's centroid should be accurate to the limits imposed by laser speckle (noise caused by interference as the laser strikes rough surfaces [8]). The use of a CMOS camera allowed fast scan rates, but introduced a sizeable FPN. This was corrected within an expected Signal to Noise Ratio (SNR) of 48 dB. The actual SNR was found to be 42 dB, which [1] indicated was probably caused by poor calibration. Evaluating the system's calibration found that the laser plane and centre of rotation were the limiting factors in the system's calibration, with the camera and lens

coefficients being acceptable. The useable range of the system in daylight was 1-2m, and 3-4m at night. The 7m range specified was not reached but this was probably due to the use of a poor stripe detection algorithm. The sensor was also vertically misaligned causing the laser band-pass filter to badly attenuate the signal, because the filter's attenuation varies based on the angle of the light striking it.

These results were not completely up to specification, but worked well considering there were several areas open to improvement. There were also other areas where the scanner could be improved. The stripe detection algorithm was slow and needed a high SNR to work limiting the range of the system. The system's calibration was not optimal and required three separate steps to complete. The platform was noisy and missed steps occasionally resulting in false position data for that scan. The PCI card limited the camera's throughput to 1MHz when it was capable of 5MHz. The attached cabling prevented full rotation of the system, resulting in a maximum angle of 180 degrees.

1.3 THE MARK-II SYSTEM

This thesis outline a second project intended to further the Mark-I scanner's goals. This scanner will be referred to from here on as the Mark-II system.

1.3.1 Its Goals

The primary aim of this project was to recreate the scanner using embedded hardware to control the scanner, read from the camera, and perform the system's processing. The use of stand-alone hardware would further the original projects goals by increasing the scanners portability, and lowering the system's cost. This scanner should be rugged and easy to move, setup, and control in the field. It should store sufficient data for practical use, and be able to transfer this data to a computer. As many of the Mark-I system's functions as possible should be implemented onboard the unit and it should operated without a controlling computer. By directly interfacing to the camera without a PCI card this system could decrease scan times.

As well as re-creating the scanner within hardware, there are several areas of further development possible to improve the Mark-I scanner, and make it more functional. The SNR of the Mark-I system was below its theoretical limit, and limited by sensor noise, so another aim of the project was to improve the SNR of the system, improving the overall accuracy of the results obtained. Firstly, alternative cameras will be investigated in an attempt to remove or lower the high FPN, suffered by the existing camera. If this is not possible improved methods of correcting this noise will be investigated and integrated into the system. Micro-stepping control for the step-motor was another suggestion in

[1] as it had the possibility of lowering the motor's excessive noise, minimising missed steps, and lowering position noise.

1.3.2 The Project's Plan

The intention was to progressively modify the system to achieve the most functional final design. In the first stage of the project the system outlined in T. Llewellynn's thesis would be re-implemented. A computer would be loaded with the original operating system, and the software created for the Mark-i system installed onto this computer. Once the unit is working, the system's performance would be evaluated against that found in the original thesis.

In the second stage the Mark-I system's low level scanner control software would be inspected, and implemented on a embedded hardware board. Input and output interfaces allowing a user to control the board would be created, and a method of communicating with computers to transfer scan data off the unit would be created. Once communications are established the Mark-I system's Java software would be modified, so that it requests scanner control functions from the the new hardware instead of the original C code.

In the final stage the Java functions for controlling the scanner, would be re-implemented on the hardware platform. This would include motor control, stripe detection, and trigonometric point calculation, as well as the sequencing of the scanning operation. Using the previous stage's computer controlled scanner, the improved features mentioned in the last section would be evaluated on the computer. If these were successful they could then be implemented on the hardware, where feasible.

Chapter 2

THE MARK-I SYSTEM'S THEORY

The properties of the Mark-I scanner were carefully designed based upon its initial specifications. These properties included the physical geometries of this components, the parameters of its optical parts, and its required dynamic range. The design of the system from [1] is described in this chapter with section 2.1 describing the calculations used to determine the orientation of the scanners parts, including working distances, and lens properties. The radiometry of the system is then described in section 2.2 by analysing the propagation of light through the system, and using this theory to determine the dynamic range required by the scanner. The dynamic range is relevant to chapter 3 where it is used to select cameras appropriate for the system, and the system geometries are used in section 6 to create triangulation algorithms that generate 3D point data, and to create the environment that simulates the operation of the scanner. The properties determined in this chapter are shown in table 2.1.

Minimum range	$X_{\min}$:	0.3850 m
Maximum range	$X_{\max}$:	10 m
Effective focal length	f_o:	8.6 mm
Baseline	b:	300 mm
Laser Projection Angle	ϕ:	68.00°
Sensor Tilt Angle	β:	1.65°
Focal length	f:	8.5 mm
Required Dynamic Range	D_P:	94.7 dB

Table 2.1 Final Mark-I system properties.

2.1 THE MARK-I SYSTEM'S GEOMETRY

2.1.1 Laser Source

The illumination source of the Mark-I system was a 5mW laser diode line generator. This was selected as a class IIIa laser to maximise the radiated power while avoiding the stringent safety precautions needed for a class II laser. This limited the laser's

power to less than 5mV in order to meet the class IIIa specification of the AS2211 standard [9].

2.1.2 Determining the Focal Length

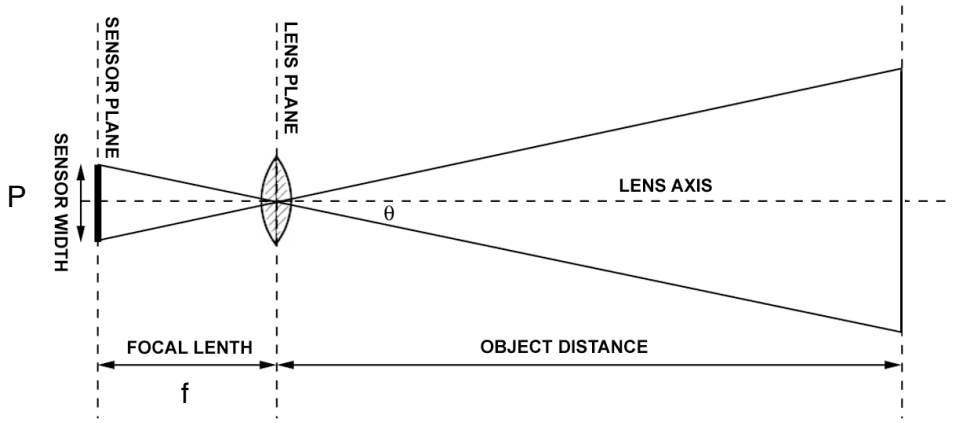


Figure 2.1 Scanner geometry that determines the system's field of view, from [1].

The Field of View (FOV) is the angle over which the system is able to image the scene. The laser line has a vertical spread of 45°, and the FOV is set as close to this angle as possible. This is determined by the sensor size (P) and the focal length (f) as illustrated in figure 2.1, and results in a FOV θ_{fov} of,

$$\theta_{\text{fov}} = 2 \arctan\left(\frac{P}{2f}\right) \tag{2.1}$$

where,

P: Sensor width (m).
f: Lens focal length (m).

The sensor chosen for the Mark-I system has a sensor size of 6.4mm x 6.4mm, which resulted in a focal length of 7.73mm for the 45° laser spread. The closest standard focal length to this is 8.5mm, which was chosen and resulted in the final FOV being 41.3°.

2.1.3 The Scanner Geometry

The system's geometry is shown in figure 2.2. A laser mounted at a baseline distance (b) from the cameras lens, projects a plane of light diagonally across the scene at an angle ϕ to the baseline. This should pass into the sensors FOV at the minimum possible distance and leaves this FOV at the maximum possible distance. The trigonometry of

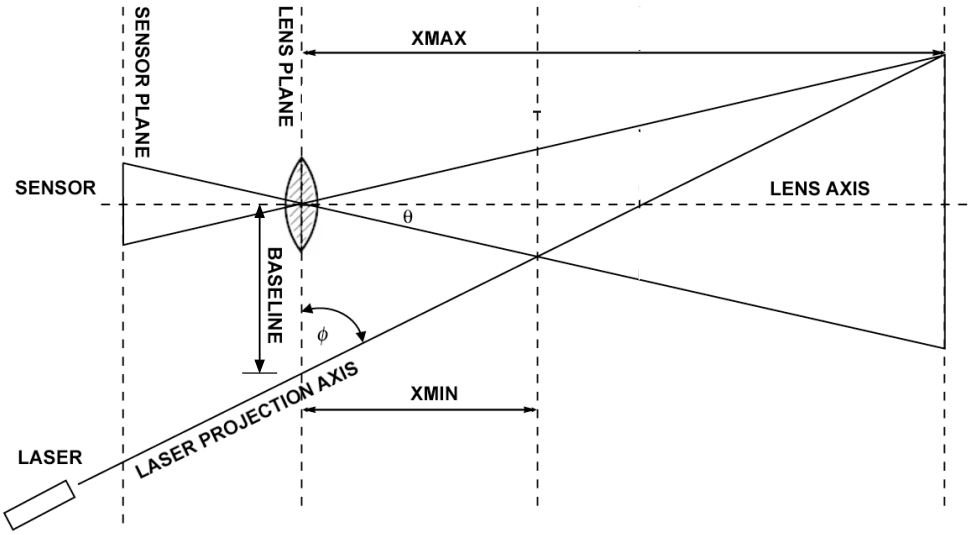


Figure 2.2 Scanner geometry that determines the scanner's range, from [1].

$X_{\min}$	$X_{\max}$	Baseline (b)	Laser Projection Angle ϕ
0.38 m	10 m	0.288 m	68.5°

Table 2.2 Object working distances, baseline and laser projection angle of system.

this arrangement uniquely determines the system's geometry from its range as,

$$b = \frac{-2X_{\max}X_{\min}\tan\frac{\theta_{\text{fov}}}{2}}{X_{\min} - X_{\max}} \tag{2.2}$$

and

$$\phi = \tan^{-1}\left(\frac{X_{\max}}{b + X_{\max}\tan\frac{\theta_{\text{fov}}}{2}}\right) \tag{2.3}$$

where,

$X_{\max}$: Scanner maximum working distance (m).
$X_{\min}$: Scanner minimum working distance (m).
ϕ: Laser projection angle (°).

The initial range from the scanners specifications was from 1m to 7m, which results in a 85cm baseline. Because this would limit the scanner's portability and increase the difficulty of its use, this range was increased to minimise b. This was done to produce a more realistic value for the baseline, resulting in the final values which are shown in table 2.2.

2.1.4 Optical Resolution

The resolving power of a lens measures the minimum distance between features on the image that can still be distinguished. For a perfect lens this limit is caused by diffraction at the edge of the aperture and is calculated as [7],

$$r_o \approx 1.22\lambda f_{\#} \tag{2.4}$$

where,

λ: the light's wavelength (m).
$f_{\#}$: F-number; the ratio of the lenses focal length to the aperture's diameter f/D.

Thus the F-number ($f_{\#}$) should be kept as small as possible, because it increases both the lenses light gathering ability and its spatial resolution. With a laser wavelength of 670 nm, and an F-number of 1.4, the minimum feature size that can be detected is 1.14 μm, which is much smaller than the pixels size.

2.1.5 Setting the Cameras Focus

The depth of view of a imaging system is the range in front of the lens, over which objects will be in focus. The lens-sensor distance and focal length determine the distance from the lens that an object is in sharp focus using the thin lens formula [10],

$$\frac{1}{l} = \frac{1}{f} - \frac{1}{f_o} \tag{2.5}$$

where,

l: Distance of an object in sharp focus (m).
f_o: Lens-sensor distance (m).

As the object moves away from this plane its image becomes more blurred, since the lens focuses light onto a point in space, rather than onto the image plane. This causes the point to appear as a blurred circle on the image plane. The range over which an object is suitably in focus can be found by determining the near (V_-) and far (V_+) limits and given by [7],

$$V_+ = \frac{d_O f^2}{f^2 - f_{\#} D_B d_O} \tag{2.6}$$

and

$$V_- = \frac{d_O f^2}{f^2 + f_{\#} D_B d_O} \tag{2.7}$$

where,

d_O (m)	V_+ (m)	V_- (m)
3.21	9280	1.61
3	45.6	1.55
2	5.30	1.23

Table 2.3 Depth of view for varying object-distances.

d_O: Object-conjugate distance (m).
D_B: Acceptable blur spot diameter (m).

which assumes that the object distance is relatively large compared to the focal length. The value of d_O was varied and the DOV evaluated resulting in the ranges in table 2.3. This showed that it was not possible to generate the maximum and minimum distances needed by the system using these optical properties.

Lowering the lenses aperture could address this, but would limit the amount of light that the lens can collect, and increase the lenses optical resolution. By changing the optical configuration of the system to use the Scheimpflug geometry [11] instead, the necessary DOV can be achieved without having to do this.

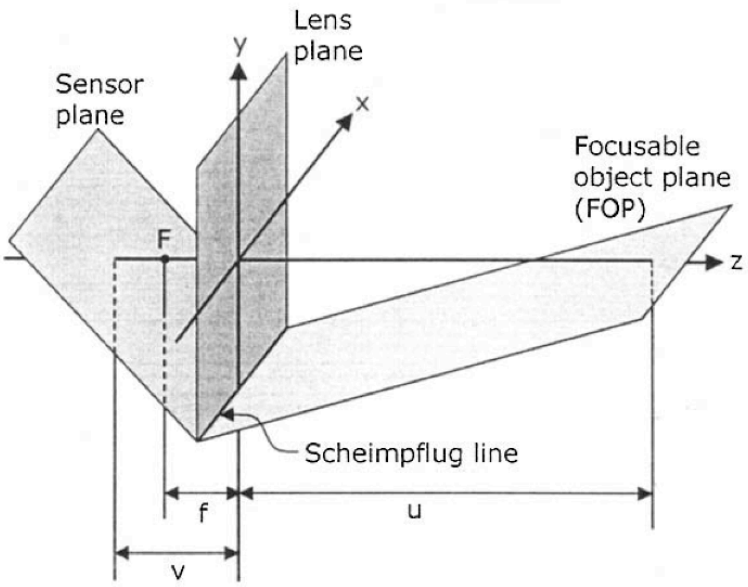


Figure 2.3 A sensor and lens in an Scheimpflug optical arrangement, taken from [11].

In Scheimpflug geometry (shown in figure 2.3) the lens or the sensor is tilted so that these are no longer parallel. This forms a plane of sharp focus which is no longer parallel to the sensor and lens planes, as it would be in a standard lens geometry. This has particular advantage for the scanner because the entire laser is now in sharp-focus, not just acceptably focused. The plane of sharp focus intersects the sensor plane at the

same place that the sensor plane does, which determines the sensor's tilt angle β as,

$$\beta = \tan^{-1}\left(\frac{f_o}{b}\right) \tag{2.8}$$

The lens-centre still obeys equation 2.5 and can be used to find the lens to sensor distance. The plane of sharp focus is then selected so that it follows the $X_{\min}$ to $X_{\max}$ ray shown in figure 2.2. Its intersection with the optical plane determines l for equation 2.5, and its intersection with the lens plane will be the system's baseline b. When the values from table 2.2 are used this yields a sensor tilt angle β of 1.65°, and a lens-sensor distance f_o of 8.599mm.

2.2 THE MARK-I SYSTEM'S RADIOMETRY

In order to understand the requirements needed for the scanner's components, the radiometry of the system needs to be evaluated. Analysing how light travels through the system allows the maximum and minimum levels of light that the system should be able to image to be found, determining its dynamic range.

2.2.1 Scene Irradiance

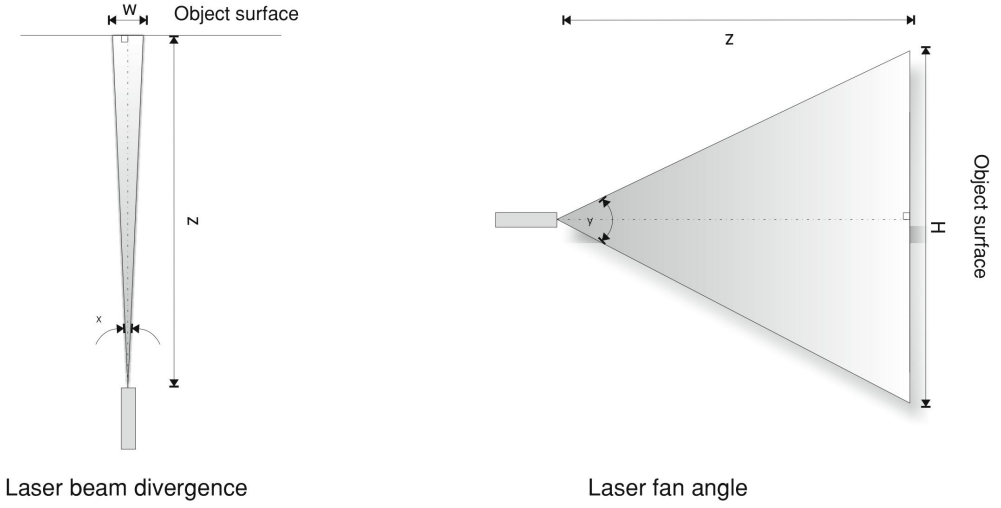


Figure 2.4 A laser diverging to fill an area on the scene, from [1].

The power of the light rays falling on a system (its irradiance), is determined by a set of optical interactions starting with the laser diode used by the scanner, which can be assumed to have a Gaussian power distribution profile [7]. The laser line generator uses a cylindrical lens to generate a line from the Gaussian laser source, and produces a line which spreads in the x and y directions and is Gaussian in both directions. The

following calculations are analysed with a non-Gaussian laser model, and correction made for this Gaussian response later.

The laser's spread is shown in figure 2.4 and produces an irradiance on the scene H_S (W/m^2) of,

$$H_S = \frac{P_t}{4z^2 \tan\left(\frac{\theta_x}{2}\right) \tan\left(\frac{\theta_y}{2}\right)} \tag{2.9}$$

where,

z: laser to scene distance (m).
P_t: laser diode optical output power (W).
θ_x : horizontal laser spread (°).
θ_y : vertical laser spread (°).

This is then re-radiated into the scene by the surface's Bidirectional Reflectance Distribution Function (BDRF). This is the radiometric measure of how a surface reflects light, and indicates how much light is reflected in each direction, given the angle and intensity of the incident light [10]. Most dull surfaces can be well modelled by a Lambertion BDRF [10], which is a distribution that reflects light so that the surface appears equally bright from all directions, and gives a radiance off the surface of N_S in (Watts/Steradian) of [10],

$$N_S = \frac{\sigma_s H_S}{\pi} \tag{2.10}$$

σ_s: surface reflectivity

The surface reflectivity varies depending on the object and has typical values between 10% and 60% [1].

This reflectance travels through the lens system and falls onto the sensor. The amount of power that is reflected off the surface and onto the sensor can be determined from the scene shown in figure 2.5, in which the lens links a cone of rays originating on a patch on the scene δO, through it's centre and onto a patch on the sensor δI. These are linked by [10],

$$\frac{\delta O}{\delta I} = \frac{\cos(\alpha_x)}{\cos(\theta)} \left(\frac{z}{f}\right)^2 \tag{2.11}$$

where,

α_x: The angle made with the lenses axis by the scene patch.
θ: The angle of the scene patches normal to its cone of rays.

The amount of irradiation projected onto the sensor by an surface patch δO can then be found using the geometry of the lens system as shown in figure 2.6. This is evaluated using the solid angle of the lens as seen from δO, and the angle that δO makes with the sensor, to give the power of the light projected onto the lens N_L as [10],

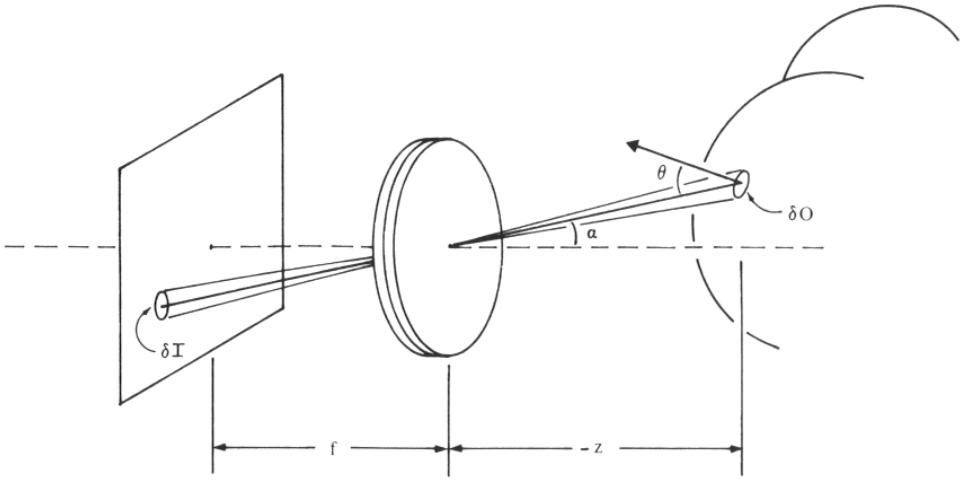


Figure 2.5 Irradiance caused on the sensor by radiance from an area in the scene being gathered by the lens, from [10].

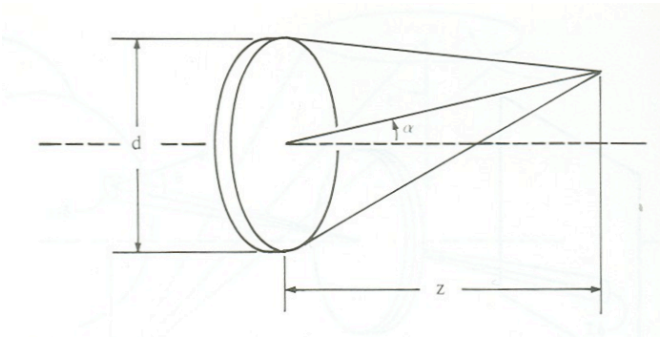


Figure 2.6 The optics geometry linking radiance from the scene to the irradiance of the sensor, from [10].

$$N_L = N_S \, \delta O \, \Delta\Omega \, \cos\theta \tag{2.12}$$

where

$\Delta\Omega$: the solid angle subtended by the lens.

Once the optical losses caused by the lens, and its filters are taken into account, equations 2.12 and 2.11 combine with the attenuation from the optics to give the sensor irradiance,

$$H_D = \eta N_S \frac{\pi}{4} \left(\frac{D}{f}\right)^2 \cos^4(\alpha_x) \tag{2.13}$$

where

η : transmission coefficient of the camera's optics (%).

To increase the system's irradiance and hence its minimum detectible illumination, the lens diameter can be increased by increasing its f-number, or decrease its focal length. It is also worth noting that as the angle the ray makes with the lenses optical axis increases, irradiance decreases in a process called vignetting.

The final received power is evaluated over the sensor array and limited by the fill factor of the sensors pixels, which is a measure of the percentage of each pixel that collects light. This results in the actual power received by the sensor being,

$$P_R = A_k \, N_S \, \eta \, \frac{\pi}{4} \left(\frac{D}{f}\right)^2 \cos^4(\alpha_x) \tag{2.14}$$

where,

A_k : Active array area, (total area $\times$ fill factor) (m^2).

2.2.2 Gaussian Laser

The laser's Gaussian response can now be factored into the system's radiometry by multiplying its transfer function by a 2D Gaussian function,

$$I(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\left(\frac{x-\mu_x}{\sigma_x}\right)^2 - \left(\frac{y-\mu_y}{\sigma_y}\right)^2\right) \tag{2.15}$$

where

σ : standard deviation
μ: mean

This is then scaled by $2\sqrt{2}\sigma$ per dimension to maintain a constant total power, resulting in a final radiometric transfer through the system P_G of,

$$P_G = 8\sigma_x\sigma_y P_R I(x, y). \tag{2.16}$$

2.2.3 Dynamic Range

Now that the transfer of light through the system is known, the maximum and minimum levels of illumination can be determined to find the dynamic range that the system needs using,

$$D_P = 20\ \log\left(\frac{P_{G\max}}{P_{G\min}}\right) \tag{2.17}$$

The Gaussian functions maximum values occur at its means $x = \mu_x$ and $y = \mu_y$, and when these are substituted into equation 2.15 the maximum power is found to be,

$$P_{G_{max}} = \frac{4}{\pi P_{R_{max}}} \tag{2.18}$$

The minimum Gaussian power is found at the extremes of the x and y values since the function continuously decays. When $x = \mu + \sqrt{2}\sigma_x$ and $x = \mu + \sqrt{2}\sigma_y$, the minimum is found to be,

$$P_{G_{min}} = \frac{4}{\pi e^4 P_{R_{min}}} \tag{2.19}$$

The dynamic range calculation $P_{G_{max}}/P_{G_{min}}$, is thus based on equations 2.9, 2.10, 2.14, with a 34.7 dB offset caused by a $20 * \log e^4$ constant. Non-varying values from these equations cancel out, resulting in a dynamic range of,

$$D_P = 20\log\left(\frac{\sigma_{s\max}}{\sigma_{s\min}}\frac{z^2_{\max}}{z^2_{\min}}\frac{\cos^4(\alpha_{x1})}{\cos^4(\alpha_{x2})}\right) + 34.7\,\text{dB} \tag{2.20}$$

The surface reflectivity can vary by an order of magnitude giving $0.1 \leq \sigma \leq 0.01$, and the triangulation angle can be safely ignored given the object's range and camera baseline. With a system range of $z = 1$ m to $z = 10$ m, the dynamic range required by the system can be evaluated as,

$$D_P = 20\log\left(10\frac{10^2}{1^2}\right) + 34.7\,\text{dB} = 94.7\,\text{dB} \tag{2.21}$$

This is an extremely large dynamic range, and complicates the selection of a camera as will be seen later in chapter 3.

2.2.4 Laser Speckle

The fundamental limit to the accuracy of active triangulation is laser speckle [8]. When a surface is rough compared to the wavelength of the laser imaging it, the light returning has different optical path resulting in interference. This appears as random bright and dark "speckle". The variation this causes in the lasers peak p was shown by [12] to be,

$$\sigma_p = \frac{1}{\sqrt{2\pi}} \frac{\lambda f_o}{D \cos \beta} \tag{2.22}$$

where

β: tilt of the sensor in a scheimpflug optical system.
λ: laser wavelength (m).
f_0: focal length.

This produces a variation of 0.38 μm for the Mark-I system (0.03 of a pixel for the 12.5 μm pixels).

2.2.5 Optical Filters

The scanner uses a polarising filter and a bandpass filter to improve the optical performance of the system. The polarisation filter blocks light of a specific polarisation to its surface. This removes the bright glare emitted from shiny surfaces that would otherwise swamp reflections from dull surfaces elsewhere in the scene. Light reflects "specularly" from shiny surfaces which preserves its polarisation [7], while dull surfaces have a Lambertion reflection, that scrambles the light's polarisation. Using these property the polarisation filter can be rotated until it is oriented to block light with the same polarisation as the laser source, removing spurious reflections.

The second filter used is a narrow bandpass interference filter, which removes the background light created by non-laser sources. This was selected so that its passbandwidth was large enough to transmit the laser beam. These filters reduce the overall transmittance of the optics by 50%, and make the system very sensitive to the input angle [7], because the light's angle of incident shifts the filters central frequency to,

$$\lambda_\alpha = \lambda \sqrt{1 - \frac{{N_o}^2}{N_e} \sin^2 \alpha} \tag{2.23}$$

where,

N_o: external medium refractive index (1.0 in air)
N_e: filter effective refractive index
θ: angle of incidence.

The system's properties of FOV = 40°, λ=670nm, and $N_e = 2.0$ produced an expected shift of 10 nm, so a filter with a centre frequency of 670nm and a bandwidth of 25-30nm was selected to allow for some variation in the system's components.

Chapter 3

SELECTING A NEW CAMERA

As part of the new project a search for a camera to improve the SNR of the overall system was undertaken, and is outlined in this chapter. The system error within the Mark-I system had been larger than expected, and the possibility of upgrading the camera to minimise the system's noise was the most obvious method of achieving this. Alternatively a new camera might offer advantages of easier hardware interfacing, over replicating the PCI interface used in the present system.

As mentioned in chapter 2 the system required a dynamic range of 95 dB or greater to image the laser stripe at both the maximum and minimum distances. This is a large dynamic range for an image sensor and requires one with a logarithmic response or specialised wide dynamic range features to achieve this.

Section 3.1 describes the properties of modern imaging sensors, section 3.3 describes the Mark-I system's camera, and section 3.4 outlines the possible replacements for this camera. The conclusion reached was that advanced linear CMOS cameras had the ability to greatly improve the overall SNR of the system, but that none of these were suitable and the project should continue with its existing camera.

3.1 IMAGING SENSORS

There are many kinds of imaging sensors available but the most commonly used sensors are CMOS and CCD sensors.

3.1.1 CCD Sensors

Charge Coupled Devices (CCDs) work by using a lattice of photo-sensitive capacitive cells that capture electrons created by the light that falls on the sensor [7]. These pixels are reset to a set constant charge at the beginning of a read cycle, and then left to accumulate charge. After a set integration period the charge packets in these pixels are shifted across the sensor to a readout-node and converted into an analogue voltage. A common amplifier connected to this node is used for all pixels. The voltages can then be converted to digital values on chip or externally.

This charge movement can be done in a number of different ways depending on the chip architecture [7]. In progressive scan CCDs the charge packets are "clocked" into the row below them. This moves the lowest row into a serial register, which then horizontally clocks the packets onto the readout node, one packet at a time.

In a Frame transfer CCD a second sensor array is created, that is covered by a photon-blocking mask. The charge is then clocked quickly from the exposed sensor array into the dummy array. This is done to prevent the charge packets from absorbing extra photons while they are moving to readout node. Interline transfer CCDs take this a step further and place these covered registers in columns between the pixels, allowing the packets to be moved into covered registers even faster. Whatever the method, the charge packets are clocked a row at a time into the serial register and then onto the readout node.

Until the last ten years CCD imagers where the sensor used for the majority of applications [13]. They are simpler to design and tend to have less noise than CMOS sensors. However, they also require a manufacturing process very different from the CMOS process, used by most foundries to make micro-processor and memory devices.

3.1.2 CMOS / Active Pixel Sensors

Active Pixel Sensors (APS) are commonly referred to as CMOS sensors because they are manufactured using the CMOS fabrication technology. They were invented around the same time as CCDs, but CCDs became the preferred sensor format mainly because they gave better image quality with the fabrication processes available [14]. As fabrication techniques improved, these sensors became realistic competitors to CCDs and began appearing in the mid-nineties. They use an array of photo-diodes or photo-gates connected to reset transistors, to detect illumination. The photo-detector is reset and then left to integrate photo-charge. This charge produces a voltage across the diode, which is then amplified by a local transistor. The amplified voltage can then be addressed by row and column selection transistors.

3.1.3 CCD and CMOS Compared

Because of the different ways in which these sensors operate, their properties tend to differ in certain areas. These differences in properties can also vary from greatly from camera to camera. This is discussed below, and some elaboration on these issues can be found in [15, 13]

3.1.3.1 System Integration and Price

CCD sensors are simpler for chip manufacturers to design, while CMOS sensors tend to have higher signal gains, and lower power requirements. CMOS sensors also allow

conventional CMOS componentry to be created on the same chip as the sensor. This allows the supporting circuitry such as clock, power, or ADC components to be included on the same chip. It also allows more advanced features such as FPN correction, or advanced processing algorithms to be performed without needing an external chip. This is especially appealing for consumer products because it allows integration of many external components into a single chip, making it cheaper to design a mass produced product around a CMOS sensor. Unfortunately this generally does not apply for high quality scientific or engineering applications where the extra cost involved in producing low noise sensors offsets this advantage. Often these CMOS and CCD sensors have similar prices. The fact that the clock, and power componentry can be located on the CMOS sensor, allows smaller connections with less propagation delay. This can allow CMOS chips to operate at higher clock rates than CCDs, although this depends on the individual camera.

3.1.3.2 Region of Interest

"Region of interest" readout is a very desirable cameras feature . It allows what is usually a square region of pixels to be read from the sensor. Rather than reading a whole frame, only the pixels of interest can be read. This allows the system to use faster tracking algorithms, speeding up the overall room scanning time. It also allows a very small region of interest to be continually re-read to create an averaged image, to reduce any random noise, such as sensor temporal noise or 50 Hz flicker from fluorescent lighting.

Because CMOS chips address each pixel, a system can read small areas of the sensor or even a single pixel, without being required to read a full frame as a CCD would. This individual pixel method may also have an unwanted side effect when fast moving scenes are imaged. CCD cameras read a snap-shot of the light at a point in time, but some CMOS sensors sample the light at different times for each pixel, leading to fast moving objects appearing to leave a "trail", across the image. Whether or not this occurs depends on the architecture used. When the architecture does not support a synchronised shutter the sensor is referred to as having a rolling shutter architecture. In these sensors the pixels do not include the extra logic required to operate a synchronous shutter for the whole sensor, which increases the amount of the pixel's area that collects light (fill factor) [15].

3.1.3.3 Fill Factor

CMOS chips have lower "fill factors", a parameter that measures the percentage of the pixel that actually collects light. Due to the extra transistors needed by CMOS architechtures, there is less room for the photo-detector on each pixels. CCD pixels use

most of their area to capture light, and thus tend to have higher fill factors, making them more efficient at converting the incident light to charge.

3.1.3.4 Noise

CMOS sensors have less uniformity of pixel response. The fact that every pixel has its own amplifier and transistors means that pixels may produce a less uniform voltage when exposed to the same illumination. This pixel to pixel difference is called Fixed Pattern Noise (FPN), and can usually be removed by arithmetic post correction. CCD sensors also have FPN but it tends to be much smaller. The extra transistors in a CMOS pixel also means that they tend to have higher temporal noise sources from transistor and amplifier noise, and from random variation in the FPN. These temporal noises cannot be post-corrected the way an FPN offset can. Blooming resistance is an advantage for CMOS sensors. They are able to drain the excess charge from an overexposed pixel, without affecting the rest of the array. This is also possible with CCDs but requires specialised architectures.

3.2 CAMERA NOISE

When examining commercial machine vision cameras the noise and performance information available is often incomplete and expressed in different measures, by different manufacturers. An understanding of how these terms relate to sensor operation is required in order to compare these differing terms.

3.2.1 Quantum Noises

Quantum noises are very important in sensors and relate to the fact that light and electricity are flows of units with a set (quantum) value. Although a flow is expressed as a continuous value, it is actually the average flow of quantal units (electrons/photons). The actual arrival of these units varies statistically [7]. That is, if an average flow in a given time is specified, the actual number of units received in a time interval will vary randomly about this average value. This variation follows a Poisson distribution, with mean arrival rate $\bar{\mu}$ of electrons (or photons) in time t. Poisson distributions can be well approximated by Gaussian responses at high μ. In these cases the noise can be classified as a Gaussian noise with average μ and standard-deviation $\sqrt{\mu}$ [7].

3.2.2 Gaussian Noise Sources

The Gaussian or Normal distribution is used widely throughout engineering and science because a large number of processes can be well approximated with a Gaussian distribution [16]. It is useful because many separate signals can be combined easily.

The sum of several independent Gaussian signals has a mean that is the sum of the Gaussian means, and its variance (standard-deviation^2) is the sum of the Gaussian variances. Thus when,

$$G(\mu_T, \sigma_T^2) = G(\mu_1, \sigma_1^2) + G(\mu_2, \sigma_2^2) + G(\mu_3, \sigma_3^2) \tag{3.1}$$

then,

$$\mu_T = \mu_1 + \mu_2 + \mu_3 \tag{3.2}$$

and,

$$\sigma_T^2 = \sigma_1^2 + \sigma_2^2 + \sigma_3^2 \tag{3.3}$$

where,

$G(\mu, \sigma^2)$:	A Gaussian distribution, defined by a μ and σ.
μ_x:	Mean of Gaussian distribution x.
σ_x:	Standard-deviation of Gaussian distribution x.

A Gaussian noise source can be thought of as a noiseless signal of value μ, with an offset added to it that has an RMS magnitude of σ. Gaussian noises sources N_{RMS}, add together to produce a signal with noise $\sqrt{N_{RMS1} + N_{RMS2} + N_{RMS3}}$.

3.2.3 Photon Shot Noise

As photons arrive at a pixel there is a statistical variation in their arrival, because of their quantal nature. Because this produces a Poisson noise it can be modelled as Gaussian with an RMS value equal to the square root of the signal due to incident light. This results in a noise with RMS value of [7],

$$\sigma = \sqrt{I \eta_{\text{qe}} t} \tag{3.4}$$

where,

I:	Flux of the incident light (photons/pixel/sec).
η_{qe}:	Quantum efficiency of the sensor (%)
t:	Integration time between the pixel being reset and its value being read (sec).

3.2.4 Dark Current Noise

Dark-current noise is the noise produced by a sensor when there is no light shining on it. It is caused by thermal energy creating mobile charge spontaneously within the sensor [7]. This happens predictably and the offset caused by this can be removed, if a reference frame of dark signal is stored. Because it is a quantal process though, this

also produces a smaller random fluctuation which is equal to the square root of the dark signal which cannot be corrected [7].

$$N_{dark} = N_d t \tag{3.5}$$

where,

N_d: The sensor's dark current constant (electrons/C°/pixel/sec).

Because the charge is generated by thermal energy, the amount of dark current increases with increasing sensor temperature, resulting in higher values for N_d.

3.2.5 Thermal Noise

Thermal noise is caused by the random fluctuation of resistances caused by the thermal energy of the system [7]. This is also referred to as Johnston noise and has the form

$$N_t = \sqrt{4kTR} \tag{3.6}$$

where,

k: Boltzmann's constant (J/K).
R: Resistance (Ω).

3.2.6 Reset Noise

Reset noise is caused by the reset transistor in both CCD and CMOS chips injecting noise onto the pixel or reset node, when it is reset. Although the noise source continues to vary, the specific value at the time of reset will stay on the pixel or node [7]. The voltage has a random variation which is similar to thermal noise.

$$N_r = \sqrt{kTC} \tag{3.7}$$

C: Capacitance that the noise is injected onto (F).

3.2.7 1/f Noise and Amplifier Noise

Both output amplifiers and photosensors suffer from a noise called flicker noise or 1/f noise. This only appears at low frequencies and increases at 3 dB/Decade [7].

Noise added by the amplifiers, tend to be a significant component of the overall noise ins sensors. This is a combination of 1/f noise, thermal noise, and inherent voltage and current noise sources within the amplifier [7].

3.2.8 Noise in CMOS and CCD Sensors

In both types of sensor photon shot noise is the largest noise source at high illumination levels. This increases with illumination level and quickly dwarfs the other noise sources. As the illumination drops to low levels other noise sources become significant.

In a CCD sensor the limiting noise sources at low light are dark-current shot noise, and the noise from the output charge to voltage amplifier [7]. In CMOS sensors the limiting noise sources are amplifier noise, dark-current shot noise, and the 1/f and thermal noise of the extra access transistors [17]. The reset noise from each pixel's reset transistor is the largest of these sources [18]. The result is that at low light levels CMOS sensors tend to have higher noise levels. CCDs have thus become the sensor of choice for sensitive scientific and astronomical applications. They can be cooled with liquid gases to reduce the dark noise below the limiting amplifier noise, to produce ultra low noise operation.

Technologies exist to lower noise for both sensors. Correlated Double Sampling (CDS) stores a reference signal (voltage or electrons) before a pixel is read, and then uses this to remove the pixel's reset noise. Multi Phase Pinning (MPP) can be used to reduce dark noise by implanting an extra positively doped region on the chip which limits its ability to create charge thermally. Both techniques can be used for CCD [7] and CMOS [14] sensors.

Although in general CCD sensors have lower noise, this can vary greatly from sensor to sensor. Differences in chip-architecture, manufacturing process, and amount of design effort spent lowering the chips noise, produce differing noise characteristics for both CCD and CMOS chips. Depending on the design and fabrication techniques used low noise CMOS chips can be produced, but this is generally harder to do than with a CCD.

3.2.9 Signal To Noise Ratio

As mentioned in chapter 2 the maximum SNR of a sensor is one of the factors that limits the scanner's range. Given above noise properties of sensors, the Signal to Noise Ratio (SNR) can be evaluated as the incident photon flux divided by a sum of Gaussian noises,

$$\mathrm{SNR} = 20log\frac{I\eta_{\mathrm{qe}}t}{\sqrt{I\eta_{\mathrm{qe}}t + N_d t + N_T t + N_r^2}} \tag{3.8}$$

This is expressed in terms of noise electrons, for a CCD which integrates and transfers electrons, but the noise figure may be expressed in volts for a CMOS sensor that converts electrons to voltage early on in its signal chain.

3.2.10 Converting Noise to Irradiation

Noise quantities are often expressed as irradiation equivalent values. These measure the magnitude irradiation required to generate a sensor signal equal to a the noise measure. This requires an understanding of how photons are converted into an electrical signal. To do this the standard equation linking received irradiation and photo-electrons generated [7] is used,

$$\text{Electrons generated/second} = \eta_{\text{qe}} \frac{P_R}{E_p} \tag{3.9}$$

where,

η_{qe}: Quantum efficiency (%).
P_R: Power of received irradiation (W/m^2).
E_p: Energy of a photon (J).

where E_p is determined by the wavelength of light used,

$$E_p = h \frac{c}{\lambda} \tag{3.10}$$

where,

c: The speed of light (m/s).
h: Planck's constant (J).

3.3 THE MARK-I SCANNER'S CAMERA

- Total dye size: 7.4 x 8.2 mm^2, 48 LCC package,
- 512 x 512 3-transistor active pixels, pitch 12.5 m, fill factor 15%,
- 6.4 x 6.4 mm optical area
- on-chip 8 bit ADC,
- on-chip multiplexing of address data,
- on-chip illumination control,
- layer of dummy pixels around the active matrix,
- pixels have a continuous operation in time (non-integrating!),
- logarithmic intensity to voltage conversion,
- extra row with reference current sources for calibration of the pixel photo currents,
- addressing speed beyond 5 MHz (X), in Y-direction: max. 500 kHz,

Figure 3.1 Fuga15 Logarithmic Camera Specifications, from [19]

The Mark-I Scanner's camera uses a FUGA15D Logarithmic CMOS sensor manufactured by Vector International [19], and its specifications are shown in figure 3.1.

The camera's dynamic range is 120 dB, and it allows a square region of interest to be read from the sensor array. Reading full frames in a darkened environment the FUGA is limited to 8 Frames Per Second (FPS), which can increase to 15 FPS in the light. Thus with 512 x 512 pixels and 8 bits the transfer rate is around 2 MB/s.

3.3.1 Camera Noise

In this camera there are two major sources of sensor noise; the CMOS sensor's Fixed Pattern Noise (FPN), and the ADC's quantisation noise [1], the largest of which is FPN. The camera's logarithmic response means that FPN correction is not the simple linear process usually required by CMOS cameras. Logarithmic cameras require a correction algorithm that is computationally intensive and is often not completely accurate [20]. The Mark-I system relied on software calibration using a curve fitting non-linear correction system to achieve reasonable results. This resulted in a noise of around 40 dB SNR, which was still worse than the theoretical maximum of 50 dB caused by the sensors 8 bit ADC quantisation [1]. Temporal noise also occurs in the system but is much smaller than fixed point and quantisation noises.

[Otim et al 2004] outlines advanced methods of FPN correction for logarithmic CMOS cameras, and these were investigated, to assess their applicability to the project. This used a standard three parameter model to correct the offset, but it was found that this deviated from the actual response at high light levels. A four parameter model which was much more complex was investigated, and found to solve this problem. The paper concludes that the three parameter method is accurate enough for all but the most stringent requirements, and simpler to compute.

The three parameter model could be applied to our system and would probably improve the corrections over the existing model. A marked effect would not be expected though, because the existing model attempts a curve fit, and is adjusted to the sensor's logarithmic curve.

3.3.2 Noise Properties

Knowing the sensor's noise equivalent irradiation power is useful for sensor comparison. The Fuga15D has a noise voltage of $V_{\text{noise}} = 0.2$ mV [1], and it's logarithmic voltage response is,

$$V_{\text{sensor}} = \gamma \log \frac{P_R}{P_{\text{limit}}} \tag{3.11}$$

P_{limit}	minimum power detectable at the sensor (W/m^2).
γ	sensor output voltage scale (mV/decade).

The noise equivalent irradiation power can be found by finding the value of P_R that gives an SNR of 1 using, [1].

$$\mathrm{SNR}_{\mathrm{sensor}} = 20 \log \frac{V_{\mathrm{sensor}}}{V_{\mathrm{noise}}} \tag{3.12}$$

When the sensors properties of $P_{limit} = 10^{-4}$ W/m^2 and $\gamma = 50$ mV/decade are used [1], this determines the sensors noise equivalent power as $P_{R_{ne}} = 1.010 \times 10^{-4}$ W/m^2.

3.4 IMPROVED CAMERA ALTERNATIVES

New cameras were investigated in the hope that the scanners precision could be increased by lowering the sensor's noise.

3.4.1 Similar Log Cameras

The Fuga-15D uses a form of CMOS sensor where the transistors driving the photosensor biased in weak inversion to generate a logarithmic response for the sensor [20]. This logarithmic response gives the sensor its unusually wide dynamic range. Similar commercial log cameras were investigated for their suitability. An example is shown in figure 3.2, and its specifications from [21] in figure 3.3.

Figure 3.2 AKATech iMVS-157 wide dynamic range camera.

IMAGE SENSOR

CMOS image sensor with 120 dB dynamics.
1024 x 1024 pixels.
Square pixels, 8µm pixel pitch.
8-bit or 10-bit greyscale.
CS-mount (C-mount with adaptor).
Spectral range from 400nm - 1000nm.

Figure 3.3 AKATech iMVS-157 specifications, from [21]

This camera is typical of the logarithmic machine vision cameras found in that it has a higher bit width, and larger pixel array, but that does not mention it noise specifications. The primary problem with the existing camera was the severe FPN limiting the systems range, and while CMOS manufacturing processes may have improved, a camera that gave information on it's FPN could not be found. The greater bit depth and pixel density may improve the scanners accuracy within its operating range, but do not address the primary problem that limited the scanner's range.

3.4.2 Linear Cameras

Most linear cameras investigated had dynamic ranges in the range of 55 dB to 65 dB, but a growing number use advanced exposure control to generate wide dynamic ranges are becoming available.

3.4.3 Basler A601f-HDR Camera

Figure 3.4 Basler A601f-HDR wide dynamic range camera.

The Basler A601-HDR [22] is shown in figure 3.4, is one such camera. It has a feature that takes multiple images at different exposures. It then combines these images as one image and outputs the data in 19 bit format, at 8 FPS. This allows both bright and dark features to be seen clearly as would happen in a log camera, creating a dynamic range of 112dB, well within that required. It is capable of reading regions of interest from the pixel array, and its specifications are shown in figure 3.5. Its linear response means that FPN correction is trivial, and it's sensor has the functionality to correct this noise on chip [23]. This system reduces both the FPN and quantisation noise sources.

The camera's properties relevant to maximum SNR, and are shown in table 3.1. When these are combined with equation 3.8, the chips SNR is determined as 46.98 dB, an improvement on the 40 dB limited Fuga-15D.

There are only two issues with the camera. The micron chip has an onboard FPN calibration feature, but the camera's documentation does not mention whether it uses

Specifications	A601f-HDR
Sensor Size (H x V Pixels)	656 x 491
Sensor Type	Progressive Scan CMOS
Pixel Size	9.9 µm x 9.9 µm
Max. Frame Rate (depends on dynamic range)	53.3 fps @ 48 dB ... 8 fps @ 112 dB
Color / Mono	Mono
Video Output Type	IEEE 1394
Video Output Format	Inside PC: 8 bit/pixel, 16 bit/pixel (linear or exponential), Laplace, Tone map, or Gamma
Power Requirements	+ 8 to + 40 VDC (12 VDC nominal), max. ~ 2 W
Lens Mount	C-Mount
Housing Size (L x W x H)	67.5 mm x 44 mm x 29 mm
Weight	max. 100 g (typical)
Conformity	CE, Fcc

Figure 3.5 Basler A601f-HDR specifications, from [24]

Quantum efficiency @ 545 nm (%)	32
Dark noise (e-)	113
Saturation capacity (e-)	50000

Table 3.1 Basler A601-HDR properties that determine its SNR, from [25].

this feature. FPN would be impossible to accurately correct off chip, because of the cameras wide dynamic range feature. Each pixel can be taken using a different exposure time, and it is impossible to know which time was used for which pixels.

The second problem is that it uses a Firewire connection, and the Mark-II system would require a Firewire host controller to be implemented in hardware. This is impractical and so a method of accessing the camera directly is needed.

Basler was contacted regarding these problems, specifically as to whether the Micron chip's FPN correction was implemented while in the cameras HDR (High Dynamic Range) mode. The response received was that it was not possible to circumvented the camera's firewire connection, and that they could not say whether the camera uses the sensor's FPN correction feature [26],[27],[28]. As a result this camera could not be used in the project, although it appeared promising.

3.4.4 SMal Camera

The other wide dynamic range camera located was the SMal ACM100 [29], and is shown in figure 3.6. Like the Basler A601-HDR , the ACM100 uses a linear CMOS sensor but generates a wide dynamic range, well within that required. It allows a sub-area of the pixel array to be read, and a development kit is also available [30].

The specifications for the camera are shown in figures 3.7. The ACM100 corrects FPN on the camera, and allows region of interest access. Its data sheet [29] does not mention the camera's bit rate but its development kit [30] can produce either 12-bit or 8-bit data. The cameras has a full frame rate as of up to 60 frames/s (fps), and the kit cam image scenes at 30 FPS. These rates are far superior to that of the Basler or

Figure 3.6 The SMal ACM100 wide dynamic range camera

ACM100 (Preliminary Specifications)	
Imager	CMOS imager with AutoBrite technology
Imager Resolution	640 x 480 active pixels
Scan	Progressive
Shutter	Rolling
Color Reproduction	Monochrome
Outputs	NTSC video
Frame Rate	Up to 60 fps at full resolution
Spectral Response	Up to 1100 nm
Dynamic Range	AutoBrite; 120 dB; adaptive and programmable
Region of Interest Exposure and AutoBrite Control	Yes: Programmable size and position
Gain	Programmable
Noise Cancellation	Yes
Dead Pixel Correction	Yes
Lens	50° HFOV lens included
Dimensions	Approximately 30 mm (d) x 33 mm (w) x 37 mm (h)
Operating Temperature	-40°C to 85°C
Storage Temperature	-40°C to 125°C

Figure 3.7 The SMal ACM100 wide dynamic range cameras specifications

FUGA cameras, and could allow advanced features to be implemented such as averaging multiple images to achieve a higher quality image.

The ACM100's properties relevant to system noise and are shown in table 3.2. Signal to noise measures could not be found for the camera, so cannot be directly com-

Noise Equivalent Irradiation	$\tilde{2}$ nW/cm^2 @ 670 nm
Minimum illumination	1 mW/m^2

Table 3.2 SMal ACM100 properties that determine its SNR, from [22].

pared with the Basler A601f-HDR. When the noise equivalent irradiation is expressed relative to metres it becomes $20\mu\ W/m^2$, much smaller than that for the Fuga-15D. This noise measure does not necessarily mean that the camera has a five times less noise that the Fuga-15D though. Because the camera's documentation does not indicate the maximum illumination its pixels can accept, its SNR cannot be determined. It is possible that this camera has a very high fill factor reducing its illumination equivalent noise. Its linear operation will still remove the largest source of noise in the Fuga-15D sensor by trivialising FPN correction.

SMal was contracted to obtain information on purchasing, sensor noise, and interfacing to their camera, but a reply was never received. Retailer for the SMal cameras or kits could not be found, and thus the camera could not be used.

3.5 CONCLUSIONS

The conclusion reached was that while new camera's were available which would increase the SNR of the system, increasing it's range and accuracy. These cameras were not suitable for the Mark-II scanner because of availability, or because of the difficulty of accessing them with our hardware. Specifically linear CMOS cameras show much promise when compared to the Mark-I system's logarithmic camera, for lowering noise and simplifying pixel correction. The prospect of creating a custom wide-dynamic range camera by directly interfacing to a sensor was rejected because of the difficulty this would entail. As a result the project continued using the existing Fuga-15D logarithmic sensor.

Chapter 4

INTERFACING TO THE CMOS CAMERA

Due to the lack of compelling alternatives at this point in time the Mark-II scanner was implemented with the existing Fuga-15D camera. This chapter describes a two module FPGA based controller and interface for CMOS cameras, that was developed. The driver module interfaces specifically to the system's Fuga-15D camera, but is shown to be easily modified to other random access CMOS sensors. Section 4.1 describes the embedded hardware used for the Mark-II scanner, 4.2 describes the timing and interface requirements of the existing Fuga-15D camera, and section 4.3 describes how this was implemented using the scanner's hardware. The remaining Mark-II system is build around this system, and if a suitable camera was available would only require the Fuga-15D driver module to be modified, for the system to function.

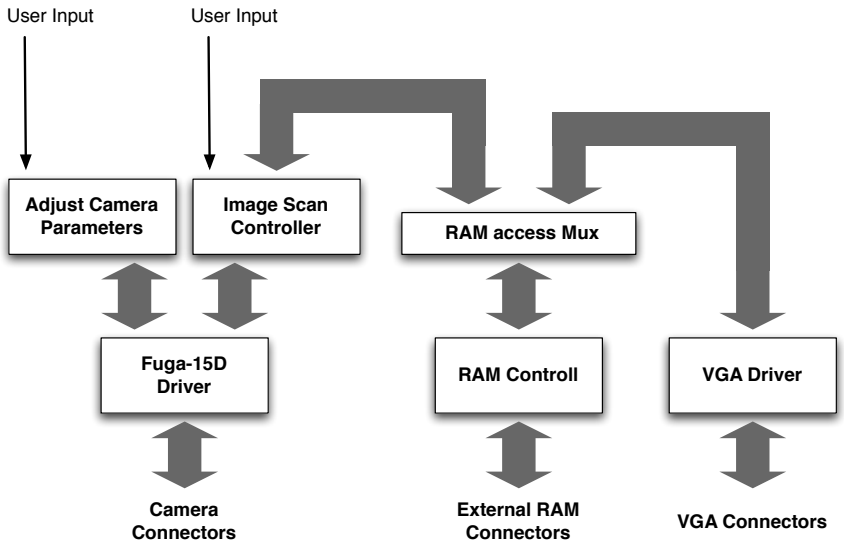


Figure 4.1 The Mark-II scanner's VHDL modules for camera interfacing.

Figure 4.1 shows the FPGA-based system that interfaces to the CMOS camera. A driver provides the interface to the camera, and is controlled by an CMOS sensor controller. This stores frames in RAM, allowing a VGA driver to display them on a standard computer screen.

4.1 HARDWARE CONTROL FOR THE SCANNER

A hardware controller was required for the Mark-II scanner, to control the platform, calculate point data, and interface to it's camera. This was to replace the processing and control performed on a PC computer in the Mark-I system.

A form of programmable logic chip called a Field Programmable Gate Array (FPGA) was chosen to achieve this. FPGAs contain arrays of logic blocks and storage elements, that are configured by software to implement state based designs. These were favoured over microprocessors for the scanner's control because they allow flexible changes to be made easily, and are well suited to high speed, high volume image processing. The ample IO resources available on FPGA allow the chips to access the system's many separate components at once, and new components to be added easily.

The configureable nature of FPGA's allow pins to be rearranged in software, and processing to be performed in parallel. Extra computation can be added without slowing existing processes, and large volumes of data can be processed concurrently. They are a common choice for system's that interface to or control machine vision cameras, such as cameras controllers ([31], [32]) or frame grabbers ([33], [34]).

In this way the whole system has moved onto a single chip, by combining the image acquisition and control and processing features, implemented separately in the Mark-I system.

The Mark-II system uses a Digilent Spartan-3 board [35] to perform its control and processing. This contains a 200,000 gate Xilinx Spartan-3 FPGA and 1 Mbyte of RAM. Connectors allow the FPGA to directly access a VGA port, three 40 pin general purpose connectors, and assorted peripherals. This board was programmed using the VHDL, Hardware Description Language (HDL), which allows device independent designs to be created for programmable logic devices.

4.2 CAMERA PROPERTIES

4.2.1 Camera Timing

[19], gives an overview of the of the Fuga-15D sensor itself, the Fuga-15D camera, and various frame grabbers compatible with the camera. It has timing information for the sensor and listed the camera's data transfer speeds.

The camera is a 512x512 CMOS array that can be addressed as a 256KByte RAM to return 8-bit values. Figure 4.2 shows the camera's timing, as outlined in [19]. The camera's latch x address (E_X) and latch y address (E_Y) lines are used to latch address data from the common address bus (DATA). A delay of at least 945 ns is needed between the end of the y-address and the beginning of the x-address on DATA. 160 ns after the address is applied the start conversion (ADCK) line can be used to start the camera's

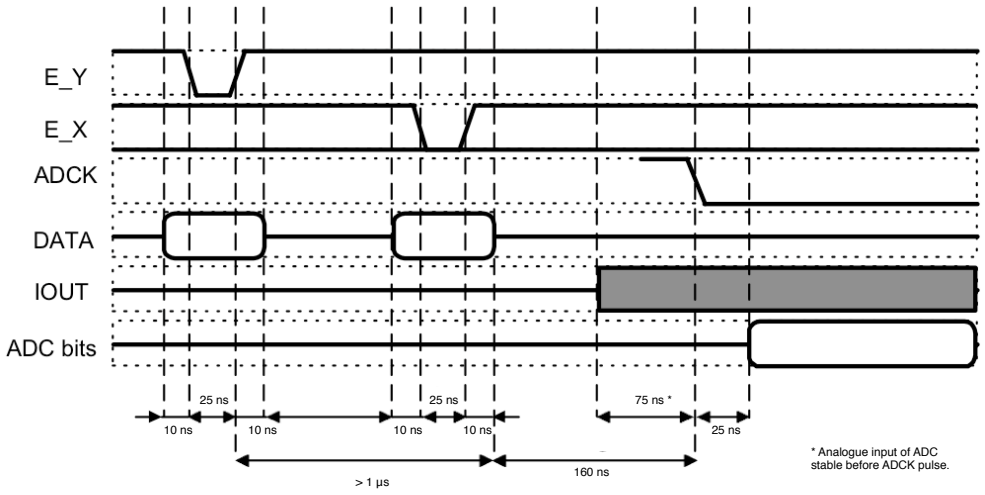


Figure 4.2 Fuga-15D signal timing for reading a pixel.

analogue to digital conversion reading a value off IOUT. The pixel's value is available on the the (ADC) bus 25 ns after this. The ADCK pin must be kept low during the readout of the ADC bits to stop the sensor's ADC constantly converting new voltages. The 945 ns delay between the x and y addresses is a result of the > 1 μs delay.

The fact that the IOUT must be stable 75 ns before the ADCK pulse is applied, has implications for the system's control. It means that ideally the camera should be stable and the laser turned on, 75 ns before this signal is applied to read the ADC. These properties result in a delay from reading a fully addressed pixel of,

$$t_{pix_{xy}} = 10 + 25 + 1000 + 160 + 25 = 1220 \text{ ns} \tag{4.1}$$

and for reading an entire row of,

$$t_{row} = 512 \times t_{pix_{xy}} = 624.640 \ \mu\text{s} \tag{4.2}$$

The actual read times are faster than this, because the y-address does not have to be set, if it is unchanged since the read. The timing for this operation is shown in figure 4.3. [19] indicates that the camera can read out 10 and 15 images a second, and generate address speeds in the x-direction of 5MHz. This corresponds to pixel-addressing speed of 200 ns, which is the timing produced by the removal of the y-address delays from figure 4.2,

$$t_{pix_x} = 10 + 25 + 10 + 160 + 25 = 230 \text{ ns} \tag{4.3}$$

This allows a entire row to be read in,

$$t_{row} = t_{pix_{xy}} + 511 \times t_{pix_x} = 118.75 \ \mu\text{s} \tag{4.4}$$

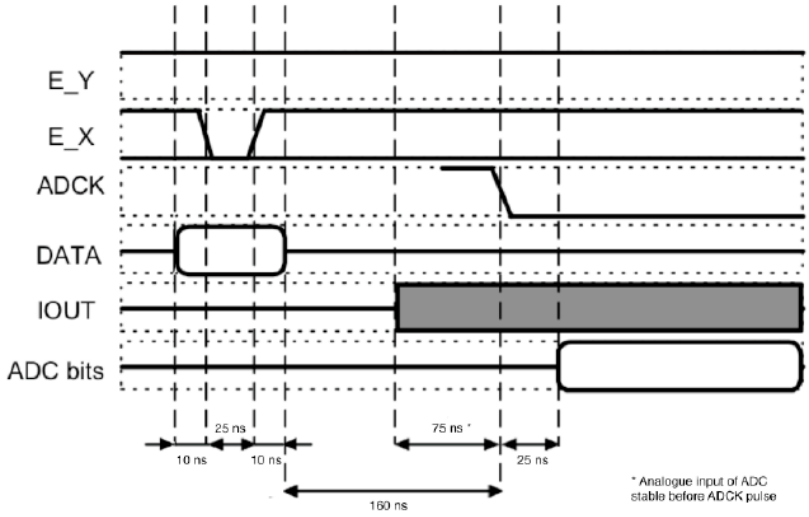


Figure 4.3 Fuga-15D signal timing for reading a new pixel, within the same row.

It also states that the camera maximum y-address response time increases in bright sunlight to 2 μs, but this does not apply to the Mark-I scanner since it images an indoor scene through a filter. The result of reading the sensor too quickly in the x-direction, is a fixed pattern appearing on the image, while reading to fast in the y-direction causes a band of over illuminated pixels to appear at the start of every row. Thus if the timing is not correct image degradation will occur.

The report also advises interfacing controllers to leave the camera's address set at a central point such as (255,255), to ensure that the automatic illumination control remains active. The result is that when a scan is started the illumination controlled ADC- bias will already be set to a useful value, avoiding saturated images because the ADC-bias had been reading a dark pixel.

4.2.2 Camera Connections

The table describing the camera's pins from [19] is included in appendix A.1. In this the ADC-bits bus is called the *adc*, and the DATA bus is referred to as *bits*. These names will be used from here on, because they are the pin names that the system's vhdl modules connect to.

Of note in this table is that the sensor's E_C pin controls access to the sensors special features, with the SNEL, and TELE lines latched on its rising edge. These lines are written-to using the *bit* pins also used to write address data to the camera.

SNEL is accessed through *bit5* and controls the automatic illumination circuit. This circuit shifts the cameras output slowly towards a reference voltage, allowing the camera to gradually adapt to bright or dark scenes. Setting SNEL low gives a fast response, and setting it high gives a slow response.

The *bit6* pin accesses the TELE line which controls the sensors reference row. When this is set high reading from row 0 returns a calibration row of current sources. When this is low the normal sensor actual row 0 will be read.

4.3 THE VHDL INTERFACE

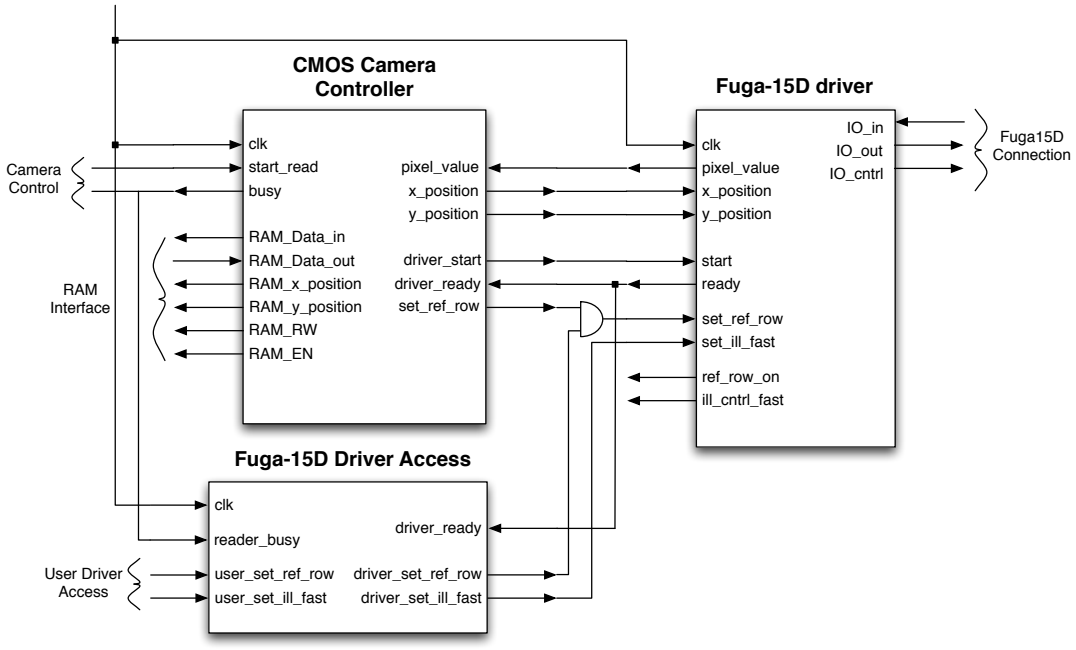


Figure 4.4 The interaction between the Fuga-15D and CMOS Camera Controller vhdl modules.

Figure 4.4 shows the VHDL modules created to interface to the camera. A low level driver provides the timing and signals required to correctly read pixels from the Fuga-15D and access its internal features. This is driven by a general purpose CMOS camera controller which reads full frames of pixels from the driver, and writes these to the Diligent board's memory for other modules to access. An extra module allows the camera's reference row and automatic illumination properties to be manually changed before a frame is read.

4.3.1 Fuga-15D Driver

The Finite State Machine (FSM) for the Fuga-15D Driver's VHDL module is included in appendix A.1. It generates the signals and timing needed to read pixels from the camera, and controls the SNEL and TELE functions. The module monitors the address sent to it and adjusts its timing depending on whether or not the y-address has changed since the last pixel was read. This allows a module calling it to ignore the specific details of the driver. The amount of delay used by the column change can be easily adjusted by changing a constant in its software, gives a 2 μs delay.

The system's FPGA board clock has a 20 ns period. This causes the timing of the signals generated to be longer than the actual delays required. For instance 10 ns or 75 ns delays cannot be created, and larger delays must be used. A board with a 200 MHz clock would be required to accurately create 10 and 25 ns states.

The FSM in appendix A.1 includes timing in square brackets, which shows the number of clocks spent in each state. Adding up these cycles the FSM takes 114 clock cycles to read a single pixel, resulting in a read time of 2.28 μs. When the y-address states are skipped this becomes 15 clocks taking 300 ns.

If the system was sped up to use a 1000 ns y-address delay the FSM could read a pixel in 64 clocks, taking 1.28 μs.

4.3.2 CMOS Camera Controller

The CMOS Camera Controller reads pixels with the camera driver, and then stores these to RAM addresses based on the pixel addresses. This starts at pixel (0,0), and reads along each of the rows until the whole array has been read. The final result is a 256K array of image data in memory.

A Fuga-15D specific version of this module was created. After reading the frame this reads the reference row and stores this in the 513^{th} row. Before returning to it's idle state, it reads pixel (255,255) to keep the automatic illumination operating while the camera is not being read.

Appendix A.1, contains the FSMs for both the generic and the Fuga-15D specific controllers. This also contains the tables A.2 and A.3, which show the time taken by each module. When this is combined with the delay times for reading from pixel's found in 4.3.1, the time taken to read an entire image frame can be found. This is shown in table 4.1.

Module	**Read-Frame Delay .**
CMOS Camera Controller	90.1 ms
Fuga-15D specific CMOS Camera Controller	90.3 ms

Table 4.1 Time taken by CMOS Camera Controller modules complete a read (20 ns clock, and 2 μs y-address delay).

4.3.3 Creating Other Drivers from the Fuga15D's

The sensor driver created for the Fuga-15D can be used as a template to design other CMOS sensor drivers. The form of this driver's read operations, which required different timing when reading pixels in the same row, compared to the timing required for those in a different row, is generically applicable to random access CMOS sensors.

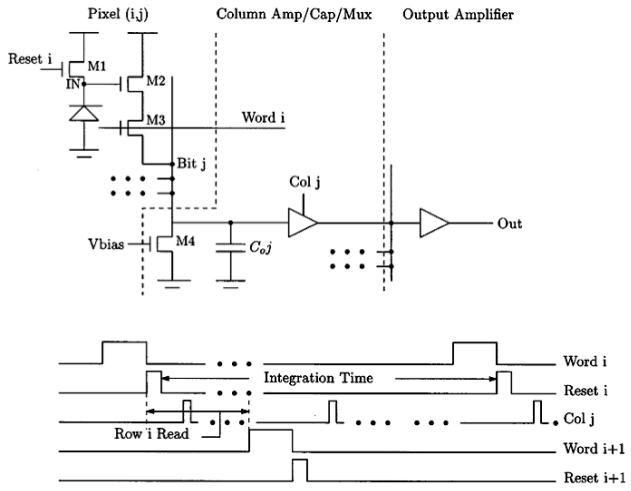


Figure 4.5 Control logic for a rolling shutter CMOS sensor, taken from [17]

As mentioned in chapter 3 the method of shuttering CMOS sensors can differ between sensors. The random access form of pixel addressing used by the Fuga-15D results in the camera having a rolling shutter. The process is illustrated in figure 4.5 showing an individual pixel, addressed by row and column logic, within a sensor array. When a sensor value is required, all the pixels in a row are reset and left to integrate charge for a set time. The charge values for the row are then stored in capacitors, which can be read out by the selection logic. Reading pixels from within the same row does not require this control sequence, but when a value in a different row is required there will be delays associated with reseting and integrating pixels.

4.3.4 VHDL System Architecture

A RAM driver and RAM bus controller are used to read and write to the Digilent board's memory and to control access to the RAM bus. The Camera Controller uses this architecture when it stores pixels, and shares this with a VGA driver. This driver accesses the bus when the Camera Driver is not operating. It reads the frames stored in RAM and displays them on a standard computer monitor, via the boards VGA connector.

Figure 4.4 shows a Fuga-15D Driver Access module. This is connected to buttons or switches on the Digilent board, and provides denounced access to the Fuga-15D drivers special features. These features are only modified when the Camera driver is inactive.

4.3.5 PCB Connector

The Fuga-15D connects to its frame grabber using a 30-wire ribbon cable, terminated with a compact female connector. It requires the cable to supply a stable power supply, otherwise darker or lighter lines may be introduced onto the image. A PCB is used to connect to this interface, in order to minimise the noise at this high speed connection. This PCB is shown in figure 4.6 , and it schematic is included in appendix.A.4. It provides a solid connection using a high quality connector, and low pass power filtering to remove the FPGA's 50 MHz clock.

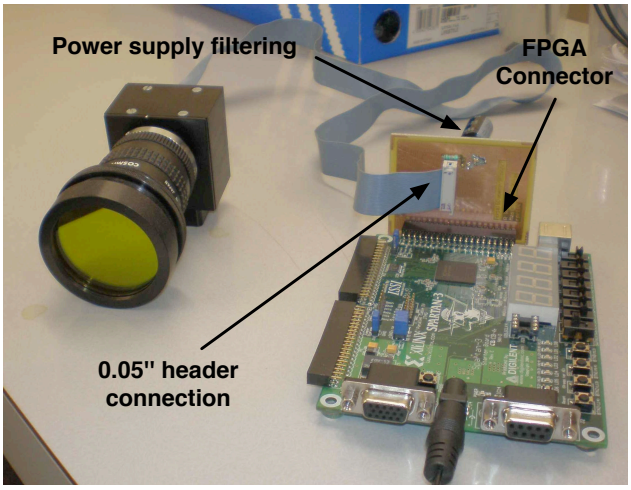


Figure 4.6 The Fuga-15D camera connection PCB.

4.4 SENSOR RESULTS

The VHDL based system was used to read from the Fuga-15D, but did not produce a useable image. The Mark-I system could not be used to test the camera because the CD containing the project's files was corrupt, so a legacy DOS driver set was used to read from the camera.

Figure 4.7 shows an image taken of the author using the DOS driver set. The noise superimposed on the image makes the camera useless for the Mark-II scanner. This degrades the system's SNR, and would prevent a laser detection algorithm from working, because the heavy salt and pepper noise would cause false laser detections.

Because the camera was unusable, the project had to be completed as a proof of concept. The Mark-II scanner uses the camera architecture developed, and reads frames from the memory in the format stored by the CMOS Camera Controller. The calculations are then performed on this data. If a suitable camera was obtained, the Fuga-15D driver module could be modified to produce a useable system.

Figure 4.7 The image read from the Fuga-15D camera with the frame grabber's DOS driver

4.5 CONCLUSION

A system has been developed to provide sensor data for the Mark-II scanner. This was implemented using the Fuga-15D because of a lack of alternatives at the time of the project, but was designed to be easily extensible to other cameras, if any are found to be suitable for the scanner. A CMOS camera controller was created to read frames of data from a camera driver and store this to RAM. This interfaced to a low level driver for random access CMOS sensors that supplied the signals needed to read pixel data from a Fuga-15D camera.

Chapter 5

3D FEATURE EXTRACTION

This chapter presents the theory involved in extracting 3D points, from the 2D pixel data received from a CMOS camera. This theory is utilised by the simulation environment and Mark-II scanner hardware, outlined in chapter 6. These combine the calculations described in this chapter with the system geometry outlined in chapter 2, to perform simulations on 3D models, and to generate 3D points from frames of camera data.

The method used to create a scan is called structured light scanning [6]. This involves projecting a controlled light source onto the scene and imaging the light reflected. This chapter describes the theory of how this works, and then outlines how this theory is implemented. Both the mathematical functions needed to solve for a perfect system, and the functions needed to improve the correction in a noisy real world environment are described.

Section 5.1 briefly outlines the problem, with section 5.2 describing the ray based approach that is used to solve this. Section 5.3 then describes the steps used to find a 3D point given a pixel value, and section 5.4 breifly outlines the method used by the Mark-I system to generate correction values.

5.1 LASER BASED ACTIVE TRIANGULATION

The task of a structured light scanner is to perform the inverse calculation of a 3D to 2D projection transform.

A ray of light has originated within the scene and been projected onto the sensor in a process described by the linear transform

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}. \tag{5.1}$$

This is not a one to one process, and thus extra information is required to perform

the inverse. This is achieved by constraining the points that can project light onto the sensor, using a plane of laser light and the pinhole camera model.

5.2 RAY BASED MODEL

5.2.1 The Pinhole Camera Model

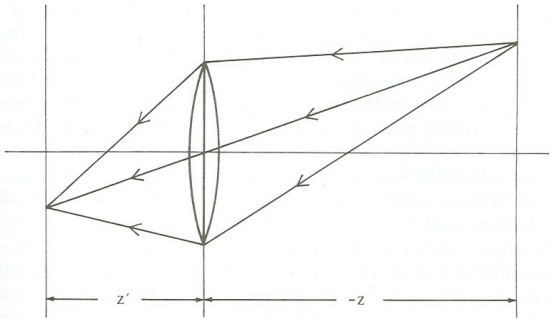


Figure 5.1 The pinhole camera model, from [10].

Figure 5.1 illustrates the standard pinhole camera model [10]. The light from an object that falls on a lens is focused by the lens onto a single point. This point forms a ray with its object, as if the object's light had travelled in a straight line through the centre of the lens. Thus the image formed by the lens appears as if its rays had travelled through a pinhole at the centre of the lens. This is provisional upon the imaged objects being in focus, with out-of-focus objects deviating from this model and blurring. Because the scanner's optical geometry was designed so that the entire range of the laser was within acceptable focus, the pinhole model to be applied to all 3D points that originated on the laser plane. A bandpass filter then blocks all non-laser light, ensuring that all features observed by the system can be modelled this way.

5.2.2 Back Projection Into the Scene

The problem of solving for the inverse of equation 5.1 is now a vector projection problem, illustrated in figure 5.2.

A ray is formed between the lenses pinhole and a 2D point on the sensor array. This is projected into the scene to find the 3D point that formed this image point. The 3D point is located at the intersection of the pixel to pinhole ray with the laser plane, and a system can now solve for the scaling value that will project this ray onto the laser plane in order to find this 3D point. The nature of the room scanner, with a rigid scanner mounted on the stepper motor's arm, means that the sensor-plane and laser-plane have a constant relative-position allowing this problem to be solved.

5.2.3 Ray Projection

The generalised case of projecting a ray onto a plane is shown in figure 5.3. A vector is formed by two points $\vec{X}_{P1}$ and $\vec{X}_{P2}$. This vector then is multiplied by a scalar u, to project it into the scene forming a new final point $\vec{X}_{P3}$. This is expressed as the equation,

$$\vec{X}_{P3} = \vec{X}_{P1} + u(\vec{X}_{P2} - \vec{X}_{P1}) \tag{5.2}$$

5.2.4 Utilising the Dot Product

The method of calculating the scaling factor u, involves utilising the three dimensional dot product $\vec{A} \cdot \vec{B}$. Aside from being fast to compute, it has a number of useful properties, including that the scalar offsets on vectors pass through the operation

$$(a\vec{X}) \cdot (b\vec{Y}) = ab(\vec{X} \cdot \vec{Y}) \tag{5.3}$$

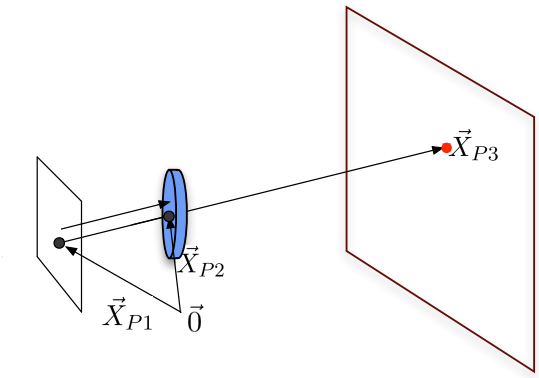


Figure 5.2 The ray formed by the camera's pinhole and a pixel intersecting the laser plane.

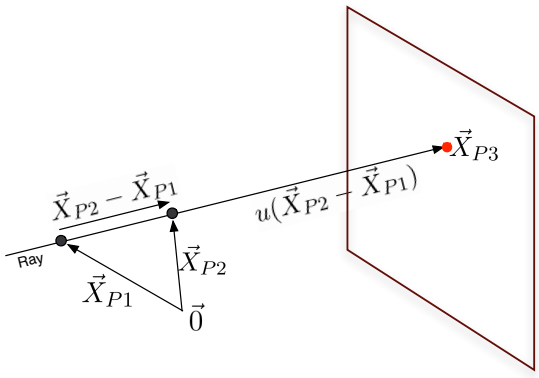


Figure 5.3 The generalised case of a vectors intersecting a plane.

and that the dot product of two vectors can also be expressed using their magnitudes, and the relative angle between them (θ) as,

$$\vec{A} \cdot \vec{B} = |A||B|cos(\theta) \tag{5.4}$$

This property, shown in figure 5.4, means that when one of the vectors is a unit vector, the dot product describes the magnitude of the second vector's component along the unit vectors direction.

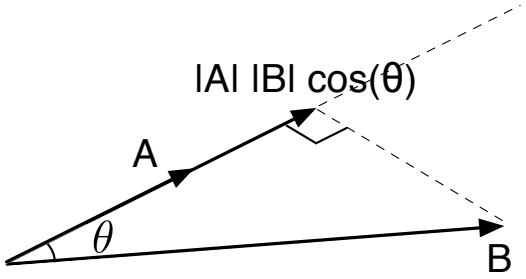


Figure 5.4 A visual representation of the dot product of two vectors.

5.2.5 Representing the Laser Plane

The laser plane can be uniquely determined from all other planes within the 3D vector space, using the planes normal vector $\hat{n}$, and the a scalar μ that projects this normal from the origin onto the plane. This can be thought of as a normal that represents the rotation of the plane, and a scalar that represents its distance from the origin. This is a convenient form of representation for the calculations because the dot product of $\hat{n}$ with any point lying on the plane will produce μ.

This is show in figure 5.5 that depicts the components of a vector that lies on the plane $\vec{X}_{Plane}$. Because the component $\vec{\sigma}$ meets the plane at a right angle it is a linearly scaled version of $\hat{n}$, and is thus the shortest possible distance to the plane. Thus,

$$\vec{X}_{Plane} \cdot \hat{n} = \vec{\sigma} \cdot \hat{n} = \mu \tag{5.5}$$

for all points that lie on the lasers plane.

5.2.6 Projecting Points onto a Plane

The value of u required to scale the vector formed by the two points in figure 5.2 onto the plane, can be calculated from

$$(u\vec{X_1}) \cdot \hat{n} = \vec{X_{Plane}} \cdot \hat{n} \tag{5.6}$$

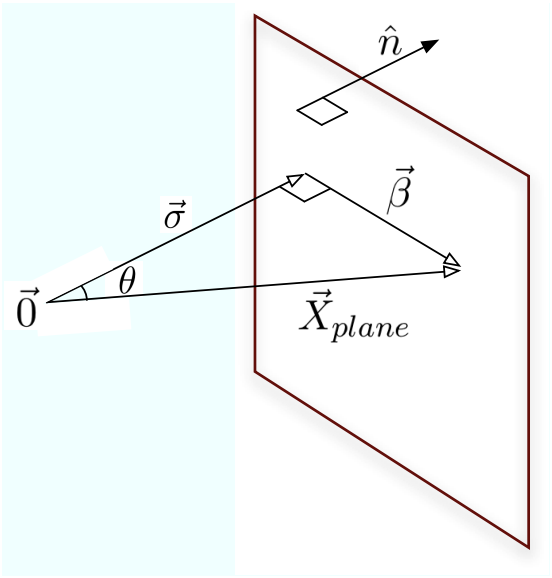


Figure 5.5 Components of any vector on a plane.

since the scalar u passes through the product,

$$u = \frac{\mu}{\vec{X_{P1}} \cdot \hat{n}} \tag{5.7}$$

5.3 SOLVING FOR THE 3D POINT

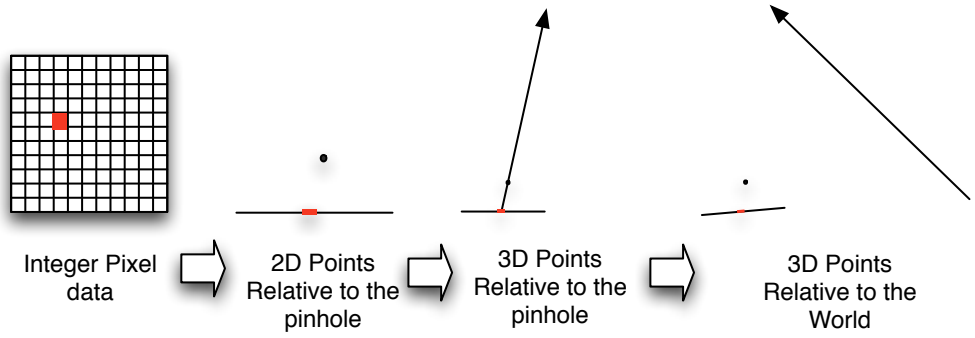


Figure 5.6 The process of generating a 3D point for the Mark-II scanner.

The process of extracting the 3D point from a 2D pixel is illustrated in figure 5.6. A pixel is converted into a 2D point relative to the system's pinhole, this is then projected into the scene, before being corrected. This process utilises the standard calibration model parameters proposed by Tsai [36]. Table 5.1 shows the parameters for this model used by the Mark-I system, which includes linear camera calibration constants, non-linear lens correction constants, and rotation and translation matrices to transform the final points relative to a final reference.

The process in figure 5.6, starts with integer pixel data x', y'. It corrects this data

Variable	Description
$\vec{X}_w = [X_w\ Y_w\ Z_w]^T$	3D world coordinate
$\vec{X}_s = [X_s\ Y_s\ Z_s]^T$	3D world coordinate rotated by the step position
$\vec{X}_c = [X_c\ Y_c\ Z_c]^T$	3D world camera
$x,\ y$	Ideal image plane coordinates
$x',\ y'$	Distorted image plane coordinates
$x_o,\ y_o$	Camera principal point
D_x, D_y	Image plane to pixel scaling coefficients
S_x	Sensor array scale factor
D_x, D_y	Image plane to pixel scaling coefficients
f_o	Effective focal length of camera
$\mathbf{R}_{cw}$	Rigid rotation between camera and world coordinates
$\vec{T}_{cw}$	Rigid translation between camera and world coordinates
$f(x)$	Non-linear mapping of ideal image coordinate x to pixel coordinates
$f(y)$	Non-linear mapping of ideal image coordinate y to pixel coordinates
$\delta_{xr},\ \delta_{yr},\ k_1,\ k_2 \ldots$	Radial distortion parameters
$\delta_{yt},\ \delta_{yt},\ p_1,\ p_2 \ldots$	Tangential distortion parameters

Table 5.1 Parameters included in the scanner model, taken from [1].

to create properly scaled 3D points located on the sensor $\vec{X}_c$. This is then projected into the scene and transformed to $\vec{X}_w$, which is expressed relative to an arbitrarily chosen world co-ordinate system.

5.3.1 Linear Pixel Correction

The images received from the sensors are merely integers representing rows and columns on the sensor. The first step in the calibration process is to remove the scaling and offset made to the actual point that the light arrived on the sensor (x, y) by storing it as a pixel, a process represented by

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} D_x S_x x f(x) \\ D_y y f(y) \end{bmatrix} + \begin{bmatrix} x_o \\ y_o \end{bmatrix} \tag{5.8}$$

D_x and D_y scale the numbers to correspond to the actual distance that a pixel causes on the sensor rather than an integer value. This means that every increment of x by 1 corresponds to an increase in x of D_x

S_x accounts for the fact that these pixels may not be square and therefore may have an aspect ration that is not unity.

Finally x_0 and y_0, adjust the pixel position so that it is relative to the sensor centre, which in this case is the point on the sensor closest to the pinhole.

5.3.2 Non-Linear Pixel Correction

The non-linear mapping functions $f(x)$ $f(y)$, values account for the distortions caused by lens aberations. These distortions can be modelled as an offset from the position the light would arrive at in a perfect system,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x + \delta_x(x,y) \\ y + \delta_y(x,y) \end{bmatrix}, \tag{5.9}$$

The Mark-I system corrected for two kinds of lens distortion, radial and tangential.

Radial distortion is caused by flawed radial curvature of lens, and stretches points towards or away from the centre of rotation. This stretching is symmetric about the optical axis and the common forms known as pincushion and barrel distortion. This distortion can be defined by the equation,

$$\delta_{pr} = k_1 r^3 + k_2 r^5 + k_3 r^7 + \cdots$$

k_1, k_2,$k_3 \ldots$: radial distortion coefficients.
r : point's distance from the image's principal point, $\sqrt{x^2 + y^2}$. (5.10)

but a reasonable correction can be made with only the second order terms, k_1 and k_2.

$$\begin{bmatrix} \delta_{xr} \\ \delta_{yr} \end{bmatrix} = \begin{bmatrix} x(k_1 r^2 + k_2 r^4) \\ y(k_1 r^2 + k_2 r^4) \end{bmatrix}. \tag{5.11}$$

The second kind of distortion commonly encountered in optical systems is decentering distortion, in which the optical centre for different lens elements are not exactly collinear.

This produces both radial and tangential distortions and can be represented by,

$$\begin{bmatrix} \delta_{xt} \\ \delta_{yt} \end{bmatrix} = \begin{bmatrix} (2p_1 xy + p_2(r^2 + 2x^2))(1 + p_3 r^2 + \cdots) \\ (2p_2 xy + p_1(r^2 + 2y^2))(1 + p_3 r^2 + \cdots) \end{bmatrix}, \tag{5.12}$$

where again a reasonable correction is achieved with only second order terms p_1 and p_2.

When these are combined as a single distortion equation, the offset caused to the an ideal point on the image plane is,

$$\begin{aligned} x' &= D_x s_x (k_2 x^5 + 2k_2 x^3 y^2 + k_2 x y^4 + k_1 x^3 + k_1 x y^2 \\ &\qquad + 3p_2 x^2 + 2p_1 xy + p_2 y^2 + x) + x_o, \\ y' &= D_y (k_2 x^4 y + 2k_2 x^2 y^3 + k_2 y^5 + k_1 y^3 + k_1 x^2 y \\ &\qquad + 3p_1 y^2 + 2p_2 xy + p_1 x^2 + y) + y_o. \end{aligned} \tag{5.13}$$

Before this model can be of any use it must invert and undistort the 2D point. The Mark-I System implemented a function approximation method which [37] had demonstrated to have a residual error of less than 0.01 of a pixel. This model was faster and more accurate than an itterative solution. It has the form,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{G} \begin{bmatrix} x + x'(a_1 r'^2 + a_2 r'^4) + 2a_3 x'y' + a_4(r'^2 + 2x'^2) \\ y' + y'(a_1 r'^2 + a_2 r'^4) + 2a_4 x'y' + a_3(r'^2 + 2y'^2) \end{bmatrix} \tag{5.14}$$

$$\begin{aligned} r' &: \quad \sqrt{x'^2 + y'^2} \\ G &: \quad (a_5 r'^2 + a_6 x' + a_7 y' + a_8) r'^2 + 1 \end{aligned}$$

A calculation of this nature, requiring fourteen additions, eighteen multiplications and a divide, is computationally prohibitively expensive for an FPGA. Given that this would have to been performed for at least one pixel in each of the sensors 512 rows, implementing this was not feasible, and the Mark-II scanner implements a linear sensor correction.

5.3.3 Generating a 3D Point from the Corrected Pixel

The actual process of correction performed by the Mark-II scanner is the inverse of the simplified sensor distortion model,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1/(D_x S_x)(x' - x_o) \\ 1/(D_y)(y' - y_o) \end{bmatrix} \tag{5.15}$$

A 3D pixel is then created to represent this corrected-pixel relative to the pinhole using the system's effective focal length f_o,

$$\vec{X}_p = \begin{bmatrix} x \\ y \\ f_o \end{bmatrix} \tag{5.16}$$

as illustrated in the figure 5.7. This 3D value $\vec{X}_p$ can then be solved for its projection onto the laser plane using equations 5.3 and 5.7. The value of u found with these then projects $\vec{X}_p$ into the scene to find the 3D point relative to pinhole,

$$\vec{X}_c = u\vec{X}_p \tag{5.17}$$

5.3.4 Co-ordinate Rotation

Once an accurate 3D point is known relative to pinhole, it needs to be transformed to the arbitrary world static co-ordinate system. This stores all of the points with

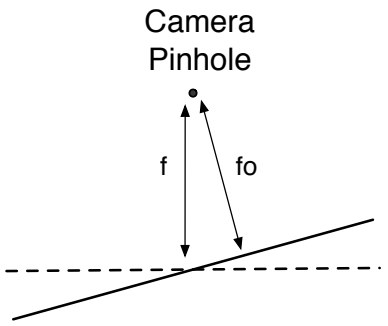


Figure 5.7 The effective focal length caused by the sensors Scheimpflug rotation.

the same rotation relative to the centre of rotation. This has two steps, involving a rotation for sensor tilt and translation to the centre of the scanner, and then a rotation to correct for the scanners step position.

First the point is rotated rotated by $\mathbf{R}_{cs}$ to correct for difference in the rotation of the sensor plane, to that of the final system. This could be any slight misalignments in a regular sensor, but it the case of a Scheimpflug geometry this corrects for its tilt. By tilting the sensor anti-clockwise the entire scene it images, appears to have rotated clock-wise to the sensor. In order to remove this rotation about the y axis, a rotation in the opposite direction to sensor tilt is required. Next a transformation $\vec{T}_{cw}$ moves the points so that they are expressed relative to the centre of the scanner, rather than the pinhole.

$$\vec{X}_s = \mathbf{R}_{cs}\vec{X}_c + \vec{T}_{cs} \tag{5.18}$$

This translates and rotates the entire co-ordinate system, adjusting the point of reference, while maintaining the relative positions of all 3D points.

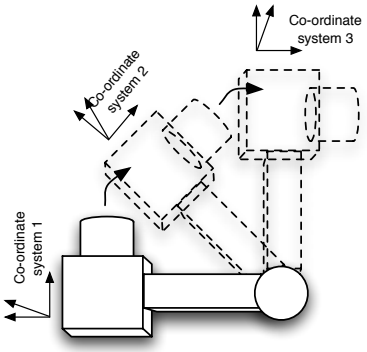


Figure 5.8 Horizontal rotation of the scanner platform.

Changes are made to the co-ordinate system by each rotation of the scanner's platform as shown in figure 5.8, so the once point has been moved to the sensor origin

the rotation caused by this process is removed with $\mathbf{R}_{ws}$.

$$\vec{X}_w = \mathbf{R}_{sw}\vec{X}_s \tag{5.19}$$

The rotation $\mathbf{R}_{cs}$ and translation $\vec{T}_{cs}$ can be combined to a 3x4 matrix by converting the 3D vector to a 4D vector, with a one in the fourth dimension.

$$\begin{bmatrix} X_s \\ Y_s \\ Z_s \end{bmatrix} = \mathbf{M}_{cs} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{5.20}$$

The scanner only rotates in the horizontal plane, so this horizontal plane is set to the centre of the scanner is to simplify calculations. A horizontal rotation in three dimensions is [38],

$$\mathbf{R} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{5.21}$$

Thus to calculate the total transformation needed to reference the data to the static co-ordinate system a matrix operation,

$$\vec{X}_w = \mathbf{R}_{sw}\mathbf{M}_{cs}\vec{X}_c \tag{5.22}$$

must be performed

5.4 FINDING THE FINAL ALGORITHMS PARAMETERS THROUGH CALIBRATION

The Mark-I System's calibration step generate the system's parameters using Matlab a toolbox based on the method proposed by [37]. Thirty two images were taken with three separate calibration targets. Using different targets allowed both larger and smaller targets to be included in the calibration, giving optimal target for both short distances. The advantages of this kind of automatic calibration routine is that the difficult task of measuring the system geometric properties such as the laser planes location or sensor orientation is avoided, and the system is tolerant to small manufacturing defects in these parameters.

The system works by expressing the same point with data from different images, using the camera parameters to link the different points. This is then re-arranged into a matrix equation, with camera parameters as the vector. An iterative error minimisation process is then performed to arrive at the set of camera parameters that most closely matches the images taken.

Variable	Description
Effective focal length	9.5838 mm
Scale factor S_x	0.9931
Principal point (x_o, y_o)	(250.7638,146.6633)
Radial distortion	$k_1 = -3.796208 \times 10^{-3}$, $k_2 = -1.099110 \times 10^{-6}$
Tangential distortion	$p_1 = 7.081575 \times 10^{-4}$, $p_2 = -3.435236 \times 10^{-3}$
Distortion-correction model a1	-0.008899
a2	-0.000061
a3	-0.000820
a4	0.003289
a5	-0.000060
a6	-0.000111
a7	0.000045
a8	-0.012889
Standard error in pixels	0.184745
Number of iterations	43
Elapsed time	9.7 s

Table 5.2 Parameters included in the scanner model, from [1].

The results of this calibration process from [1] is shown in table , and has a static transformation matrix of,

$$\mathbf{M}_S = \begin{bmatrix} 0.9987 & -0.0029 & -0.0480 & -141.9042 \\ -0.0396 & 1.0219 & -0.0210 & -32.2576 \\ -0.0302 & -0.0035 & 0.9914 & 1.2903 \end{bmatrix} \tag{5.23}$$

If a new camera is selected this process can be repeated by manually reading images, and re-running the toolbox, to generate new values for the camera properties.

Chapter 6

THE MARK-II SCANNER

This chapter describes the implementation and testing of the FPGA based Mark-II scanner. A system for controlling the hardware and calculating results was created by adding additional units to the camera controller system described in chapter 4. Because a working camera was not available, a simulation environment was created to evaluate the operation of the Mark-II system. Both systems are based on the theory outlined in chapter 5, and the Mark-I's scanner's properties described in section 2.

Section 6.1 outlines the design and implementation of the various units required by the scanner, and section 6.2 describes the structured light simulation system developed in Matlab.

6.1 THE FPGA CONTROLLED SCANNER

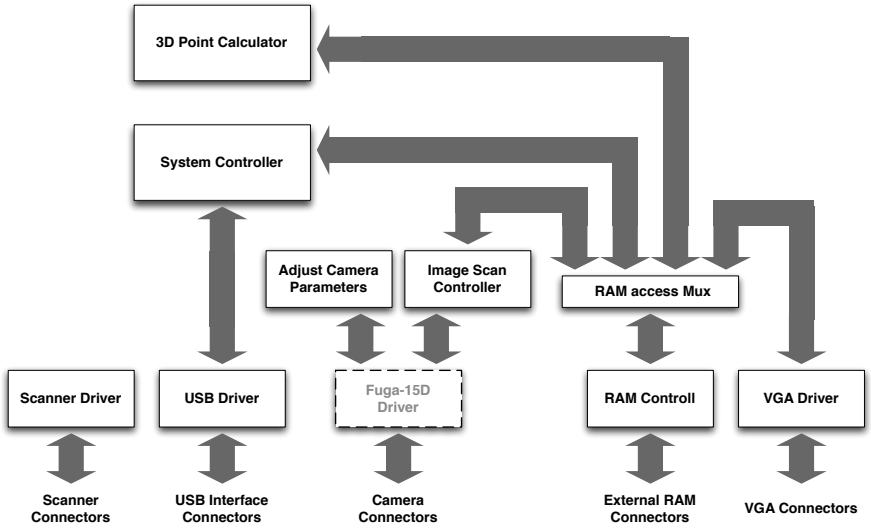


Figure 6.1 The Mark-II scanner's VHDL modules.

Figure 6.1 shows the VHDL design modules created for the final Mark-II scanner. A RAM based approach was used in which the design modules share the same RAM space. This allows large blocks of data to be easily shared and operated on while

requiring minimal interaction between components. The sequencing of the system's operations is controlled by the System Controller module, and a user can access the scanner using simple IO, or its USB connection.

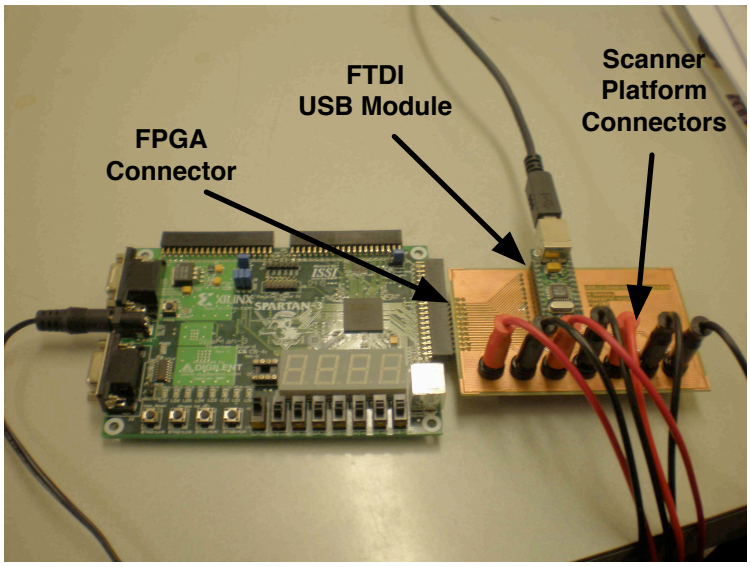


Figure 6.2 The Mark-II scanner's circuit board.

Figure 6.2 shows the circuit board created to allow access to the scanner's hardware and to the PC. This houses the connections for the step motor, laser and USB communications circuitry, and its schematic is included in appendix A.5.

6.1.1 System Controller

The System Controller module is the main VHDL unit in the Mark-II scanner. This controls the other modules, and the sequence in which operations are performed on the scanner. It allows a user to control the unit through the FPGA board's IO, and also allows requests to be sent to the scanner via USB. This unit allows blocks of RAM to be written or read manually which was utilised to test the system's point calculations.

The system's RAM architechture is shown in figure 6.3. All of the system's modules operate on or read from either the image frame block of memory or the calculated point block of memory. Because it gives access to the system's RAM through the USB port, the System Controller allows a testing system to place a simulated image frame directly into memory, as if it had been read there by the Image Scan Controller. This is then operated on by the 3D Point Calculator which stores its corrections to RAM in a separate block, that can be read using the System Controller. This method was used to verify the operation of the Mark-II scanner, using a Matlab simulation environment described in section 6.2 below. Simulated image frames were created and then processed by the Mark-II scanner, and the results evaluated against the original data.

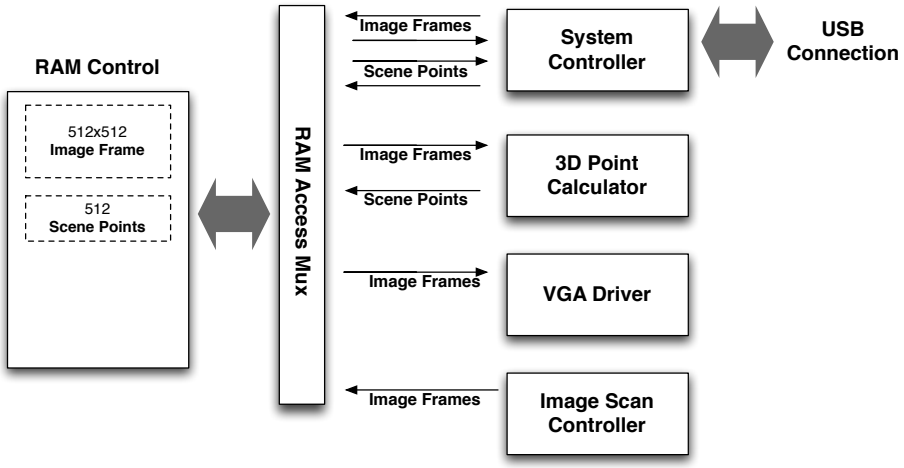


Figure 6.3 The Mark-II scanner's memory architechture.

6.1.2 USB

Although designed to run without a computer, the scanner requires a connection to a PC to transfer images to a computer in case of testing or calibration, and to transfer the point data it generates to a PC. USB was chosen to accomplish this because of the ubiquity of USB connectors on modern computers, and reasonably high data rates these support.

An FTDI USB module [39] which allows parallel communication by the hardware to travel over a USB bus, is used to accomplish this. Drivers provided by FTDI then converts this connection to a serial port inside the computer. This system is capable of transferring data from the scanner at 300 KB/s [40], and works under Macintosh, Linux and Windows operating systems. Speeds of 1 MByte/second are also possible but require a separate FTDI driver that limits the system to the Windows operating system. This is configured by the circuit board in figure 6.2 to operate in a 3.3V bus powered mode.

6.1.3 Scanner Platform Control

The system's scanner drivers operate the rotating platform and controls the laser, replicating the functionality of the Mark-I system's control drivers. It turns the laser on and off, rotates the motor in step, and half step modes, rotates continuously to a given position, locates the motors home position, and continuously evaluates whether the motor's position is correct. This has a maximum step speed of 250 Hz and an angular resolution of 0.125°.

A step motor is used to position the scanner. These are motors manufactured with set "step" positions that the motor rotates to. These positions are reached by

trying to precisely align magnets within the motor, making these motors simple to control digitally. The motor rotates the scanner through a two cog gearbox in order to increase its resolution, by reducing the size of each step.

[1] suggested the implementation of a micro-stepping algorithm, which would use complex current and voltage waveforms to position the step-motor to "micro steps" which are smaller than the step resolution. It was suggested that this would lower the motors audible noise and minimise position-error. This is not possible because the motor's external control pins do not give direct access to the motor windings. The complex waveforms required for micro-stepping cannot be applied without opening the motor's case to access these windings, which was not attempted because AC voltages are contained within the housing.

6.1.4 3D Point Calculator

A calculation unit was created that converts pixel data to 3D points, utilising the structured light triangulation model described in chapter 5. FPGA support for arithmetic operations, and the arithmetic modules used are described in sections 6.1.4.1 and 6.1.4.3, and the calculations themselves are outlined in section6.1.4.4.

6.1.4.1 Arithmetic

The computational capabilities of FPGAs are very flexible, because of the lack of a fixed architecture on the chip. This allows variable bit widths, and components that operate in parallel without blocking. The downside to this is that common mathematical functions may not be implemented, and must be created. The increased flexibility also comes with an increase in the design complexity. Rather than specify a set of sequential processor commands a Finite State Machine (FSM) must be specified to sequence the commands, and transfer inputs and results between registers.

Xilinx FPGAs use the XST synthesis tool [41] to implement HDL designs such as VHDL. This converts the design into lower level structures that can be implemented onboard the FPGA. It supports native addition, subtraction and multiplication in VHDL, but can only implement the divide and remainder operations, when the divisor is known at compile time and is a power of two.

6.1.4.2 Multiplication

Spartan-3 FPGAs also provide Embedded Multiplier modules, which are implemented in the chips silicon rather than created using its logic resources [42]. These are used within the scanner's computation module to minimise the logic it requires. These multipliers can perform 18x18 bit signed, and 17x17 unsigned multiplication, and perform

the 18 bit multiplication in 5.81 ns [43]. The multipliers can be configured to calculate fewer bits, which decreases this delay.

Utilising multiplier modules also had the advantage of hiding its shift operations inside the multiplier module. Half of the multipliers result cannot be stored and is discarded, which appears as a division by a power of two for unsigned values. Its affect on signed values is erratic and may not actually divide the number by a power of two. In order to implement a arithmetic shift (one that always divides by a power of two), algorithms must convert the number to a signed value, shift this and then convert it back. The multiplier implements this arithmetic shifting internal to the module returning the shifted result, and hiding this complexity from other parts of the design.

6.1.4.3 Division

In order to perform the calculations in chapter 5 a custom division module was required. The implemention of division is common problem for FPGA arithmetic. It is the hardest of the basic arithmetic operations to implement, and FPGAs tools usually have little or no support for the operation. Division algorithms for hardware tend to have long propagation delays and high resource requirements [44].

These algorithms utilises recursive multiply and shifts operations to solve,

$$\frac{N}{D} = q_0 + q_{-1}\frac{1}{2} + q_{-2}\frac{1}{4} + q_{-3}\frac{1}{8} + \dots \tag{6.1}$$

for a given number of quotient bits q_{-1} to q_{-n}. The algorithms are limited to cases where $N < D$, ensuring a fractional result with $q_0 = 0$, and can be rearranged to form,

$$D(0.1011011\dots) = N \tag{6.2}$$

Shifting a binary number results in a "partial remainder",

$$D(1.011011\dots) = 2N \tag{6.3}$$

This determines the value of q_{-1}, based on whether $D \leq 2N < 2D$ (), or $0 \leq 2N < D$. If the new q_0 term is 1, it is removed by subtracting D from the partial remainder, so that the next q_i can be found. This shift-subtract process is repeated recursively until the required number of quotient bits have been found.

The two common methods of implementing this algorithm on FPGAs are restoring and non-restoring division. Higher Radix, and SRT division are also commonly used algorithms that have smaller propagation delays, but are not suitable for FPGAs because they greatly increase the resources required [44].

[Bailey 06] evaluated various configurations of restoring and non-restoring algorithms on an FPGA using a high level C-style language. These algorithms are de-

scribed in appendix A.6, and compromise equations A.1– A.4. They were re-created as VHDL modules performing 8 bit division, and their resource requirements and timing delays were compared, with the results shown in table 6.1. The logic requirements are measured in "Slices" which are the modules used by the FPGA's Combinational Logic Blocks to implement logical functions [43]

Division Algorithm	**Slices**	**Longest Propogation Delay (ns)**
Restoring algorithms		
A.1	103	42.585
A.2	73	52.385
Non-Restoring algorithms		
A.3	122	48.366
A.4	58	46.767

Table 6.1 Delay and resource comparison of division algorithms implemented in VHDL on a Spartan-3 FPGA.

[Bailey 06] found that the non-restoring algorithms were faster and used fewer resources than the restoring algorithms. Among the non-restoring algorithms it found that A.3 required fewer resources, while equation A.4 had a shorter propagation delay. The results from implementing these in VHDL showed that the restoring algorithm A.1 had the shortest delay, while the non-restoring algorithm A.4 used the least resources. The reason that these implementation results vary is hard to evaluate, given that a different compiler and FPGA combination were used in the two cases.

Efficient resource utilisation was a higher priority in the Mark-II system, so algorithm A.4 was used. Extra logic prior to the algorithm modifies the inputs so that they are positive, and these are shifted so that $N < D$. These changes are removed from the result by similar logic after the operation completes.

6.1.4.4 The 3D Point Calculation Module

Figure 6.4 shows the calculation process performed to by the 3D Point Calculation Module, with the constants uses are included in appendix A.7.

The calculation values are all multiplied by powers of two, because the FPGAs fixed point system cannot compute fractional values. To represent a fractional number, its value is stored multiplied by a power of two, to produce an integer. Thus x (5 >>) indicates that the actual value stored by the FPGA is x $\times 2^{-5}$.

Power of two scaling allows numbers to be expressed with much greater precision, but requires that a calculation's sequence is carefully considered. Multiplication and division will operate seamlessly on shifted numbers,

$$732 \times 2^{16}/(62 \times 2^4) = 732/62 \times 2^{12} \tag{6.4}$$

```
(>>7)        (>>7)           (>>7)
temp_a = pixel_p_x - fpga_pixel_offset_x
temp_b = fpga_pixel_offset_y - pixel_p_y

(>>40)         (>>7)       (>>33)
point_2d_x = temp_a x fpga_pixel_width
point_2d_y = temp_b x fpga_pixel_width


(>>39)  (>>22)         (>>17)
mult1 = point_2d_x x fpga_laser_norm_x
mult2 = point_2d_y x fpga_laser_norm_y
mult3 = point_2d_z x fpga_laser_norm_y

(>>21)         (>>21)  (>>21)  (>>21)
add_result  = mult1 + mult2 + mult3


(>>-1)        (>>18)                (>>19)
div_result = fpga_laser_mag_r / add_result

div_result = div_result <<7


(>>28)         (>>22)          (>>6)
point_3d_y = point_2d_x x div_result
point_3d_y = point_2d_y x div_result
point_3d_z = point_2d_z x div_result


(>>26)  (>>16)          (>>10)
mult1 = fpga_mat_11 x point_3d_x
mult2 = fpga_mat_12 x point_3d_y
mult3 = fpga_mat_13 x point_3d_z

(>>8)            (>>8)    (>>8)   (>>8)   (>>8)
final_point_x  = mult1 + mult2 + mult3 + fpga_mat_14


(>>26)  (>>16)          (>>10)
mult1 = fpga_mat_21 x point_3d_x
mult2 = fpga_mat_22 x point_3d_y
mult3 = fpga_mat_23 x point_3d_z

(>>8)            (>>8)    (>>8)   (>>8)   (>>8)
final_point_y  = mult1 + mult2 + mult3 + fpga_mat_24


(>>26)  (>>16)          (>>10)
mult1 = fpga_mat_31 x point_3d_x
mult2 = fpga_mat_32 x point_3d_y
mult3 = fpga_mat_33 x point_3d_z

(>>8)            (>>8)    (>>8)   (>>8)   (>>8)
final_point_z  = mult1 + mult2 + mult3 + fpga_mat_34
```

Note:

(>>32)
number_a

Inidicates that the actual value of number_a is,

number_a / 2^32

Figure 6.4 The 3D Point Calculator Module's calculations including rounding, with the 2^x offsets indicated.

but addition and subtraction must be performed using numbers with the same arithmetic offset,

$$3 \times 2^{14} + 4 \times 2^{13} \neq 3 \times 4 \times 2^{x} \tag{6.5}$$

This means that the sequence of calculations must be organised so that the operands for any additions or subtractions have the same offset.

The calculations performed by the 3D Point Calculation Module were carefully planned to maximise the values entered into the calculations while ensuring common scaling offsets at the addition stages. The module also performs post calculation shifting, to ensure that its results do not overflow the 18 bit data set. Every multiplication discards 18 bits, implementing a 2^{18} division, and every four bit addition discards 2 bits, dividing these results by four.

The 3D simulation environment described in section 6.2 below, allowed the calculations to be further optimised. This allows the maximum values of interim variables that cannot be known at compile time, to be considered. The module performs division using a value determined at calculation time. The 3D simulation environment allowed the range of possible values that this division result could take to be determined. In the calculations shown in figure 6.4 the scaling of the division result by 2^7 will not overflow because the largest result that the division can produced given the current geometry is -1226. This can be safely stored with an offset of 2^6, allowing a scaling of 7 to be applied to maximise the precision of this number.

6.2 SCANNER SIMULATION

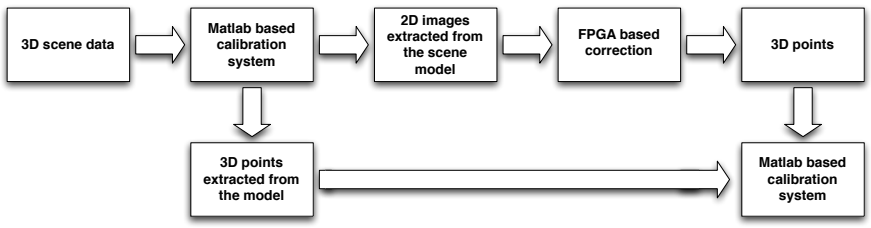


Figure 6.5 The Mark-II scanner's VHDL modules.

A simulation environment for the scanner system was created in the Matlab environment. This creates images and point sets, that are used to evaluate the system, and test the Mark-II scanner's embedded hardware. Figure 6.5 shows the flow of data through this environment. 3D data models are converted into Matlab data, which is then simulated using the structured light triangulation theory outlined in chapter 5 to produce virtual pixel images that would be formed by a camera in the system.

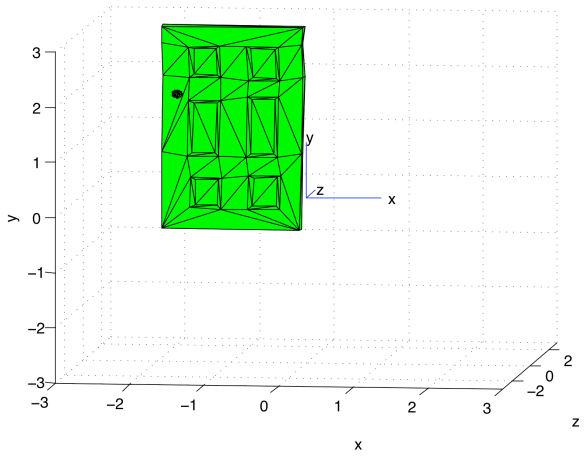


Figure 6.6 A 3D Matlab scene imported from an ASE file.

6.2.1 Extracting 3D Data

The 3D data is created by importing triangular meshes from the ASCII Scene Export (ASE) 3D data format using a script. This script automatically generates Matlab scripts that can be executed in Matlab to declare the triangular data contained in the ASE file. Figure 6.6 shows an example of an ASE data set that has been imported into Matlab. ASE is a native format of the popular 3D graphics application 3D Studio Max. This program allows other 3D data sets to be transformed into the ASE format for use within the simulation environment.

6.2.2 Simulation Environment

The simulation environment's geometric properties are shown in figure 6.7. Its specific parameters were selected based on the geometry from table 2.1 in chapter 2, and the sensor properties of the Fuga-15D camera [19].

The system works by creating a set of rays along the laser's plane which pass through the system and form points on the sensor, a process shown in figure 6.8. These rays are extended into the scene until an intersection with a triangle from the data set is found. A second ray is then formed with this intersection by extending it through the lenses pinhole and onto the sensor's plane. This point is then scaled and offset based on the sensor's dimensions to obtain the final pixel value. This is rounded to an integer and stored in a virtual image frame. The system's geometries correctly account for the sensors Scheimpflug tilt, and the model ignores points that are outside the bounds of the sensor. The final 3D pixels are then sent to the Mark-II scanner, 3D point corrections generated and the data compared against the original 3D points found by the simulator.

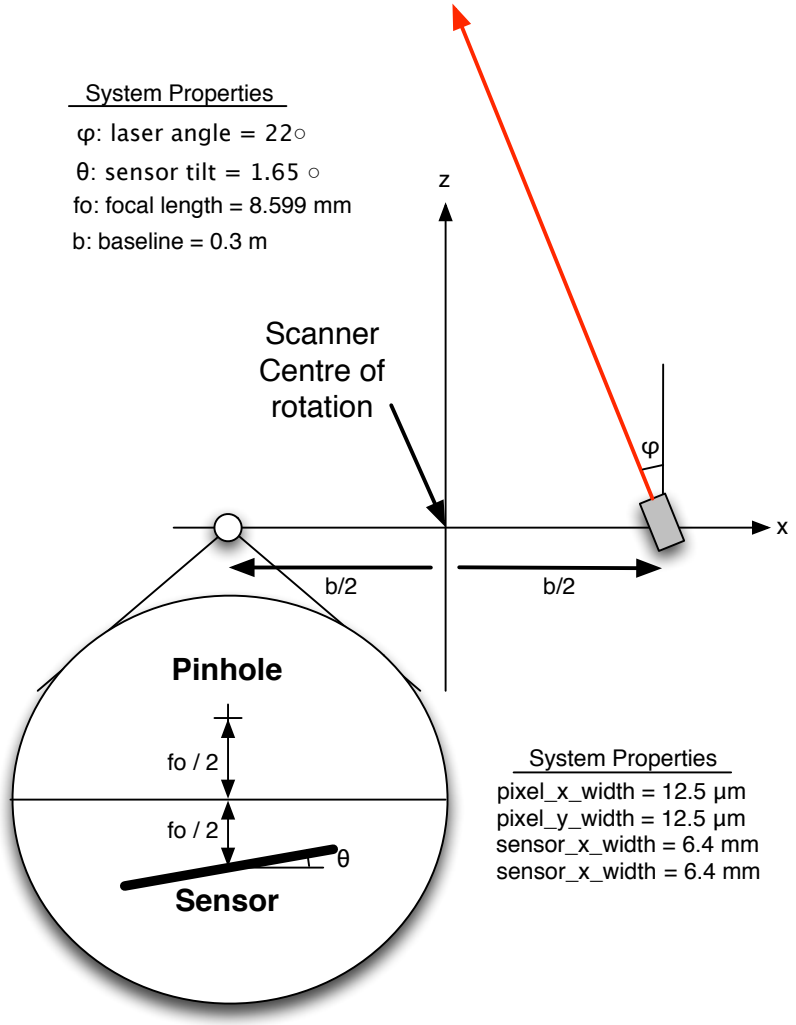


Figure 6.7 The Mark-II scanner's simulation environment.

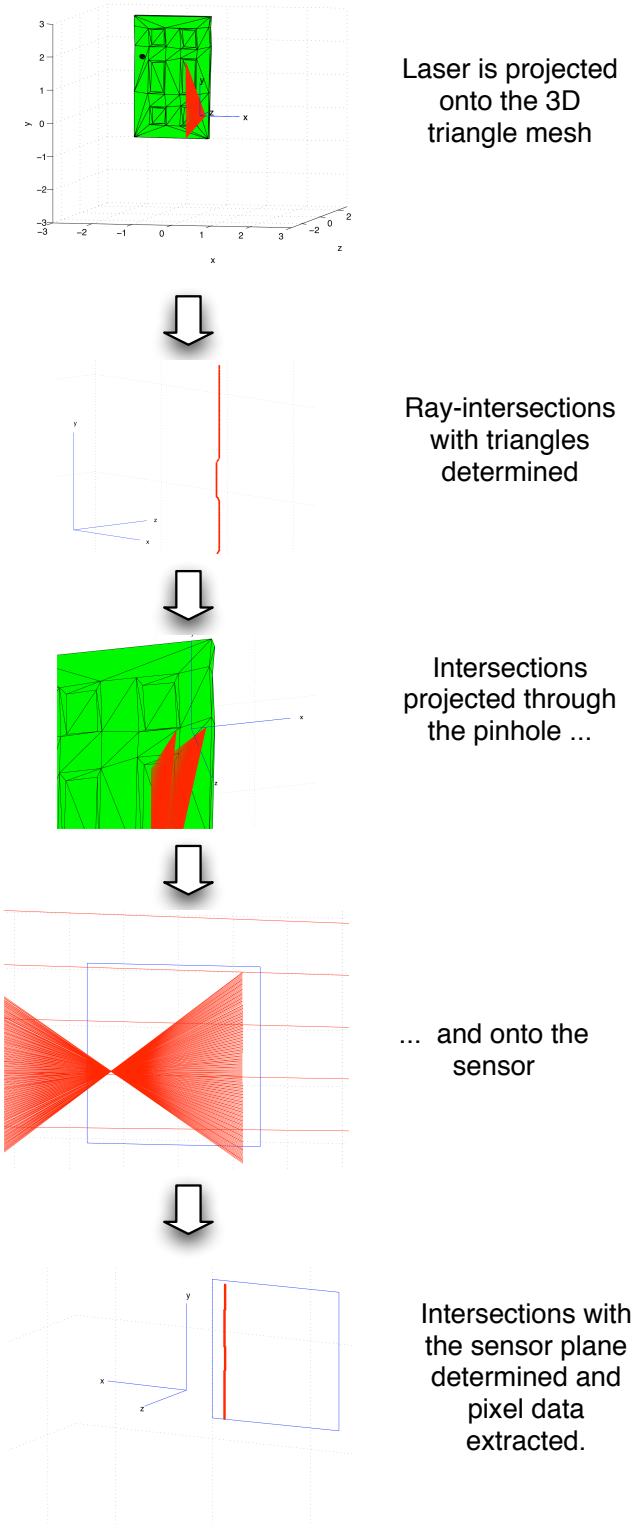


Figure 6.8 The simulated environment projecting laser rays off a 3D scene and projected onto the sensor, in order to create a frame of pixels.

Chapter 7

CONCLUSIONS

This thesis outlines a project to re-implement a 3D structured-light triangulation scanner. This system, referred to as the Mark-I scanner, was designed to digitise interior rooms by creating point maps, and was optimised for these conditions. This project performed its calculation and control using computer software, and the new project's primary objective was to implement this functionality using embedded hardware to make the scanner smaller and more portable. A secondary goal of this Mark-II scanner was to improve the SNR of the system by selecting a newer machine vision camera.

An FPGA based design specified in VHDL was created that re-implemented the scanner in embedded hardware. This controls the scanner's platform, reads images from a CMOS camera, and generate 3D point data from image frames using structured light triangulation, all without the need for a PC. It is able to transfer its calculated points, and image frames to a PC via USB connection. A simulation environment was created to evaluate and test this scanner. It is based on the same structured light model, and imports data from a 3D data format.

Chapter 1 introduced the Mark-I scanner and discusses the motivations and objectives of the Mark-II scanner, chapter 2 gave an overview of the Mark-I system's design and properties, chapter 3 evaluated the possibility of replacing the scanners sensor, chapter 4 described the VHDL system created to interface to the system's camera, chapter 5 described the theory required to operate a structured light triangulation scanner, and 6 described the final system created.

The high precision and wide operating ranges required for an indoor scanner meant that a very wide dynamic range was required. This resulted in the Mark-I scanner using a logarithmic-response sensor in order to operate over the required ranges of illumination. Unfortunately noise reduction for logarithmic cameras is difficult, particularly for Fixed Pattern Noise (FPN) a point observed in the limited performance of the Mark-I scanner.

As part of the re-implementation process a search was conducted for cameras containing the required wide dynamic range, and offering lower system noise. This determined that logarithmic response sensors are no longer required to obtain the scanner's

wide dynamic range. New linear CMOS sensor architectures are capable of producing these ranges, resulting in increased SNRs because of simplified noise correction and the greater bit depths theses cameras provide. The requirement for the system to interface to theses cameras with embedded hardware was a limiting factor though, and meant that an improved camera was not available for the Mark-II scanner at the time of its implementation.

The project continued utilising the existing logarithmic camera. A generalised architecture for camera access was created, and was designed to be adaptable to other CMOS sensors. This was implemented using an FPGA programmed using the VHDL language. A RAM based system architecture was designed on this board to evaluate this camera, by reading and transferring image frames to RAM, and then displaying these frames on a computer monitor using the VGA format. This determined that the camera was faulty and the project would have to be completed without a functional camera.

In order to allow this, a simulation environment was created. This imported standard 3D data sets, and used this model and the system's geometries to generated virtual sensor frames for the scanner to read.

The Mark-I scanner's motor and laser control where then re-implemented on the embedded hardware and a USB connection added to the system. A point calculation unit was created that uses the perspective projection model to create its 3D points. This was designed using the simulation environment to maximise the precision of its fixed point calculations. It reads image frames from the area of RAM where the driver module stores its data, generates 3D points for the pixels and then stores the final points to a separate location in RAM. A board controller was created that operated the scanner and controlled its modules.

This allows a user to issue requests, and to read and write to the board's RAM from any PC program with access to its computer's serial port. This was utilised with the simulation environment to evaluate the Mark-II scanner's 3D point calculation module, by manually loading data into the RAM space where pixel data is written to, and then reading the data back from the area of RAM where the result is stored.

7.1 FUTURE WORK

The Mark-II scanner can be improved in a number of ways, to add new features to the scanner, better evaluate its performance, and improve its effectiveness at calculating points.

- The system can be upgraded to use an FPGA with more processing resources. The current board uses an affordable, medium to low density FPGA, and FPGAs with higher logic density are readily available. The system can be ported

to these easily, since it is specified in VHDL with a minimum of device-specific components used. The scanners multiply module is the only device-specific component. It should be easily replaceable on another FPGA architecture, and can be synthesised in logic if required. The current multiply module can also be used on Xilinx's high density Vertex-II FPGAs without modification.

- A new camera can be selected to implement the working system. The number of linear CMOS wide dynamic range cameras on the market is increasing indicating that a suitable camera may be available in the near future.

  When selecting a new camera careful consideration should be given to device interfacing, as most machine vision cameras are accessed by either the ethernet or camera link protocols, neither of which are suited to FPGA control.

  During the course of the project the camera manufacturer SMal was unable to be contacted regarding its cameras, but the ACM100 evaluated was recently found to have been acquired by Cypress Semiconductor Inc. who have now on-sold the camera to Sensata Technologies. This camera may now be available for use and any future scanner projects should contact Sensata Technologies to investigate its availability [45].

- The system currently lacks the storage required by a working scanner to store the large of data created by scanning a 360° room at every step position and a solution is required for this. Hard drive storage would be ideal because of the extremely large storage capacities it affords, but is ill-suited to an embedded system. Commonly available hard-drives require PC bus connections such as ATA, and generally do not provide information on accessing the drives outside of these complex standards. A more realistic solution is to supply the required storage using Flash memory chips. These storage devices are cheap and starting to rival small hard-drives in their storage capacities. The amount of data storage needed by the scanner can vary depending on the final application. If the system calculates points to within the theoretical sub-pixel limit found in [1], then 141 MB would need to be stored, while the existing system would require 4.2 MB to store a full 360° scan.

- With a functioning camera it would be possible to manually capture and download a set of images for the calibration system to examine, but ideally this would be implemented as an automated feature that prompted the user at every step and stored the image frames separately from the rest of its data.

- The fixed point calculation algorithms can be further altered and evaluated, to determine the optimal shifting scheme. The present algorithm ensures that none of its calculations overflow, but the majority of the points it calculates will be much lower than this overflow maximum. The scanners calculation unit can be

optimising for the average case and any overflows clipped to the maximum value. The average error could then be compared to that of the existing system to evaluate whether or not this improves the scanner's overall precision.

- The conversion of the calculations to 38 bit computation is also possible by chaining together two multiplier modules, and updating the bit widths of the other calculations. This will improve the systems precision, but may well require an upgrade to an FPGA with more resources.

- Multiple scan merging is a desirable feature for the room scanner. This takes multiple scans from different locations and attempts to fit them together by minimising the reprojection error between the scans. This would require a recursive error minimisation algorithm that would be very computation intensive, and may well be best evaluated on a PC using scans stored by the portable scanner, and then ported to hardware if this was practical. Utilising a marker such as a cube of checker boards, that would be common to the separate scans would decrease the computation required and simplify the algorithm.

- A continuing project could evaluate new motors. Direct access to the motors windings, or existing micro-stepping functionality would be desired to further increase the system's resolution. Light weight or lower power usage would also be desirable. The lack of information on the existing motor meant that evaluation of the battery requirements for a fully portable scanner were not possible. Selecting a new motor could allow the feasibility of this to be determined.

- The lack of a camera prevented the range and performance of the system being properly evaluated. A future system with a working camera could evaluate these by scanning small, precisely known objects (such as a flat plane), at many different locations within the scanner's range. This would determine how its localised error varied across its range. A much lager object of known shape, such as a long cylinder, could then be used to evaluate the gradual systematic error, that might for instance gradually curve the viewed points across the scene.

- The Mark-II scanners point sets can be viewed using its Matlab simulation environment, but this proprietary software may not be available on all systems, and method of viewing the data downloaded from the scanner would be useful.

- Neural Networks are a form of advanced computer computation, which could be specifically useful for the system. This method involves teaching a network of interconnected neurons to recognise patterns, and provide a result based upon the training sets they have observed. This could be particularly useful for minimising the computation involved in large calculations such as the lens-distortion

correction implemented in the Mark-I system. These networks view all functions as patterns, rather than a set of commands, making this large function as computationally expensive as any other. Training these networks can require a lot of processing, but can be achieved on a computer, and then the constants transferred to the actual network onboard the FPGA.

Appendix A

APPENDIX

A.1 PIN CONFIGURATIONS FOR THE FUGA-15D CAMERA

Pin	Name	Main function	Remarks
1	GND	Camera ground (0V)	Reserved for future use
2	GND	Camera ground (0V)	Reserved for future use
3	bit1	Address data	
4	bit0	LSB of address data	
5	bit3	Address data	
6	bit2	Address data	
7	bit5	Address data	set automatic. illumination control - see E_C and SNEL
8	bit4	Address data	
9	bit7	Address data	activate 2x2 pixel grouping - see E_C and 2X2RGB
10	bit6	Address data	activate calibrated sources on row 0 - see E_C and TELE
11	E_Y	Enable Y-address register (active low)	
12	bit8	MSB of address data	
13	TRIH or MPX	Tristate control of 4 MSB of ADC output, or ADX multiplex control	is MPX in releases after June 96. Normal use: MPX=1.
14	ADCK	Clock of ADC. Data ready shortly after falling edge	
15	TRIL DIO	tristate control of 4 LSB of ADC output, or test diode	is “DIODE” (or not used) in releases after June 96
16	E_ADC	General tristate control of ADC (active low)	
17	E_X	Enable X-address register on input data. This is an upgrade compared to earlier versions of the LPT1-card (Active low)	tied to E_X in releases after june96
18	E_C	Enable registers of control inputs (active low)	the control bits activated by E_C are common with the address bits.
19	adc1		
20	adc0	LSB of ADC output	
21	adc3		
22	adc2		
23	adc5		
24	adc4		
25	adc7	MSB of ADC output	
26	adc6		
27	VDD	power supply (5V)	
28	GND	camera ground (0V)	
29	EGC	External gain control	
30	GND	camera ground (0V) Reserved for future use	

Table A.1 Pin configuration of the Fuga-15D camera taken from [19].

A.2 FUGA-15D VHDL MODULES

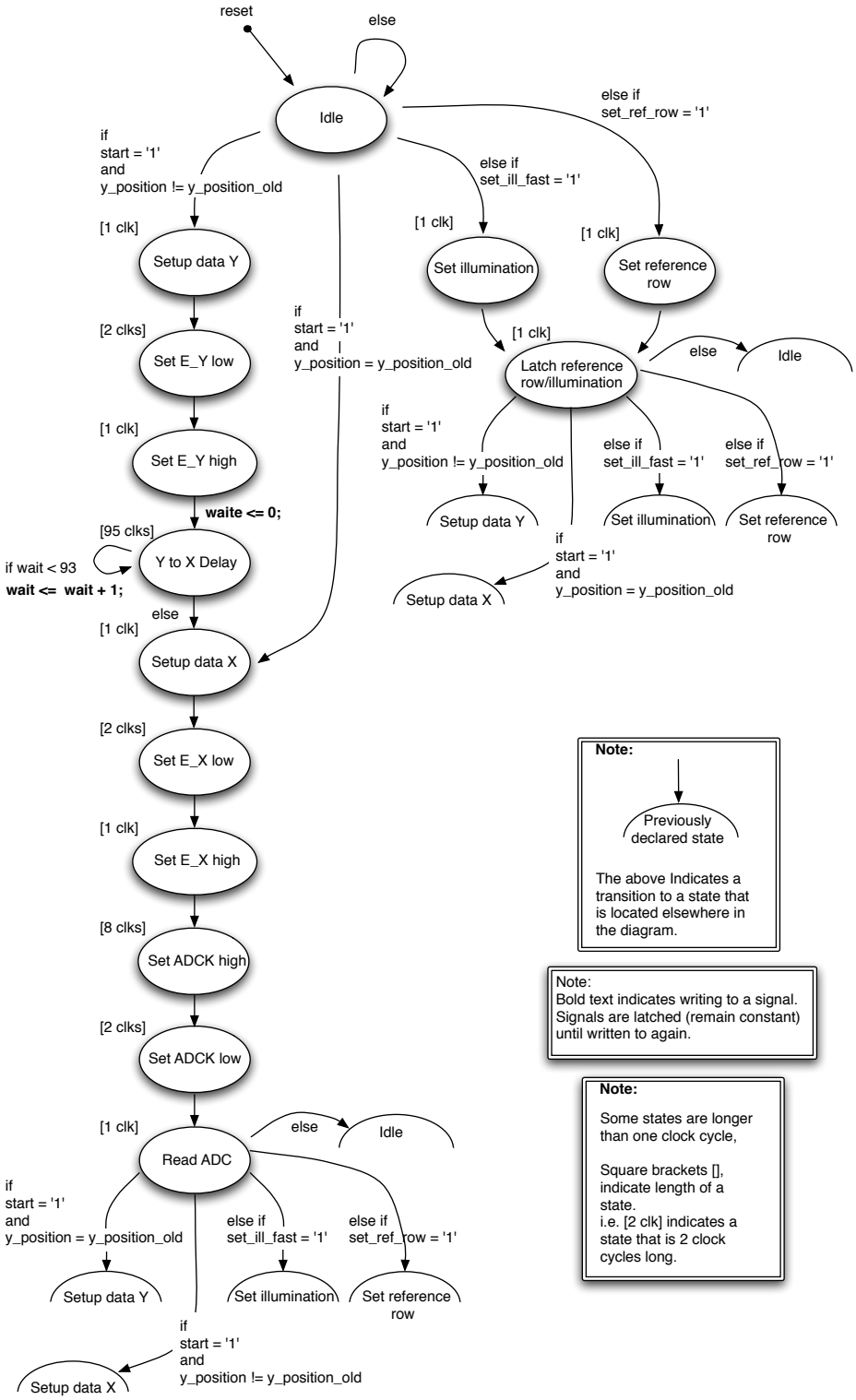


Figure A.1 Finite state machine for the Fuga-15D driver VHDL module.

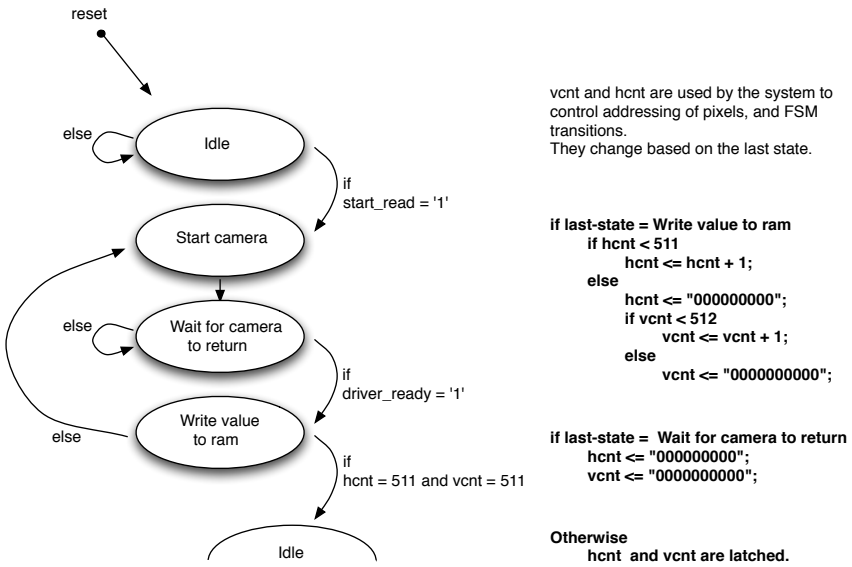


Figure A.2 Finite state machine for the CMOS Camera Controller VHDL module.

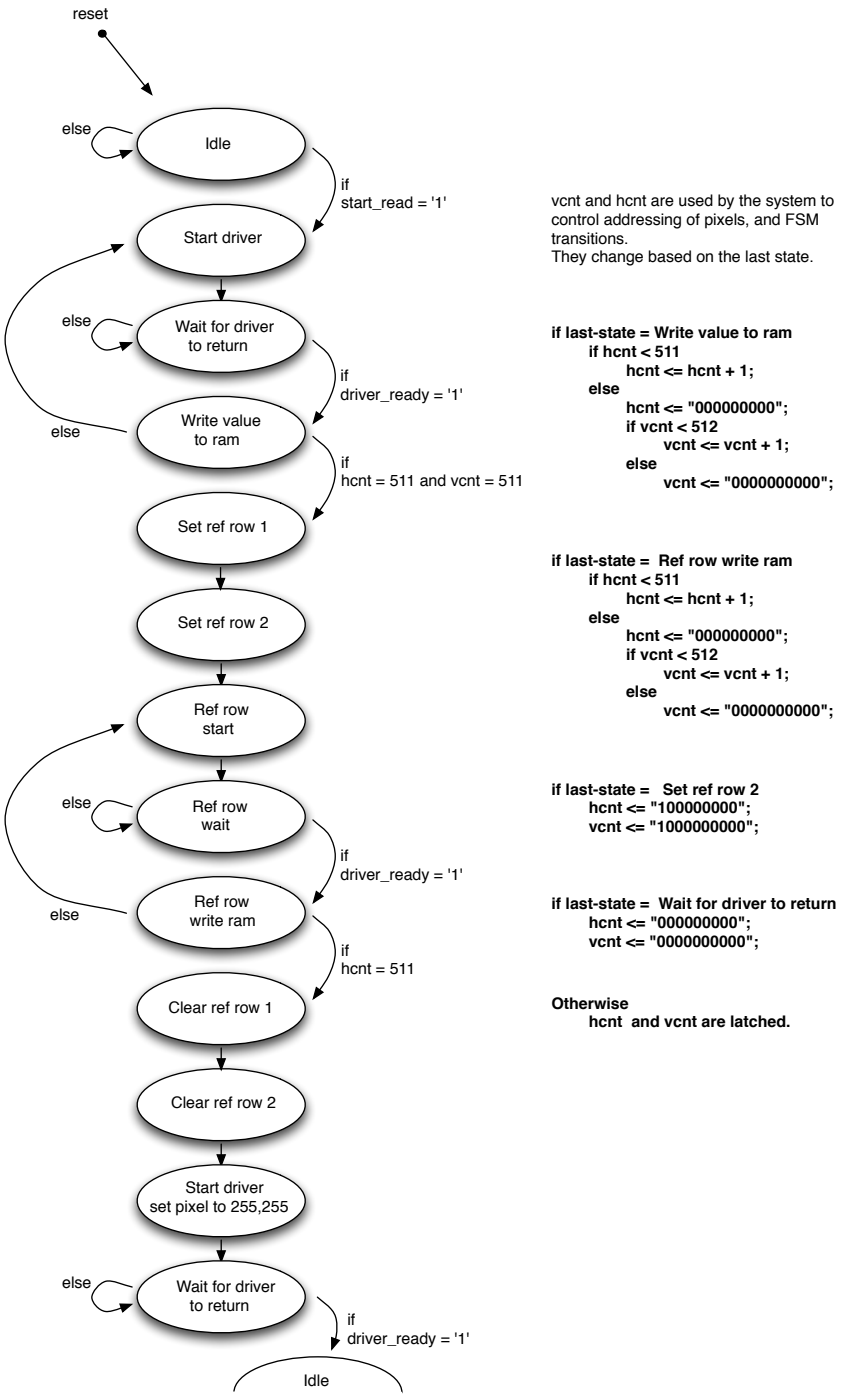


Figure A.3 Finite state machine for the Fuga-15D specific CMOS Camera Controller VHDL module.

A.3 READ TIMING FOR THE CMOS CAMERA CONTROLLER

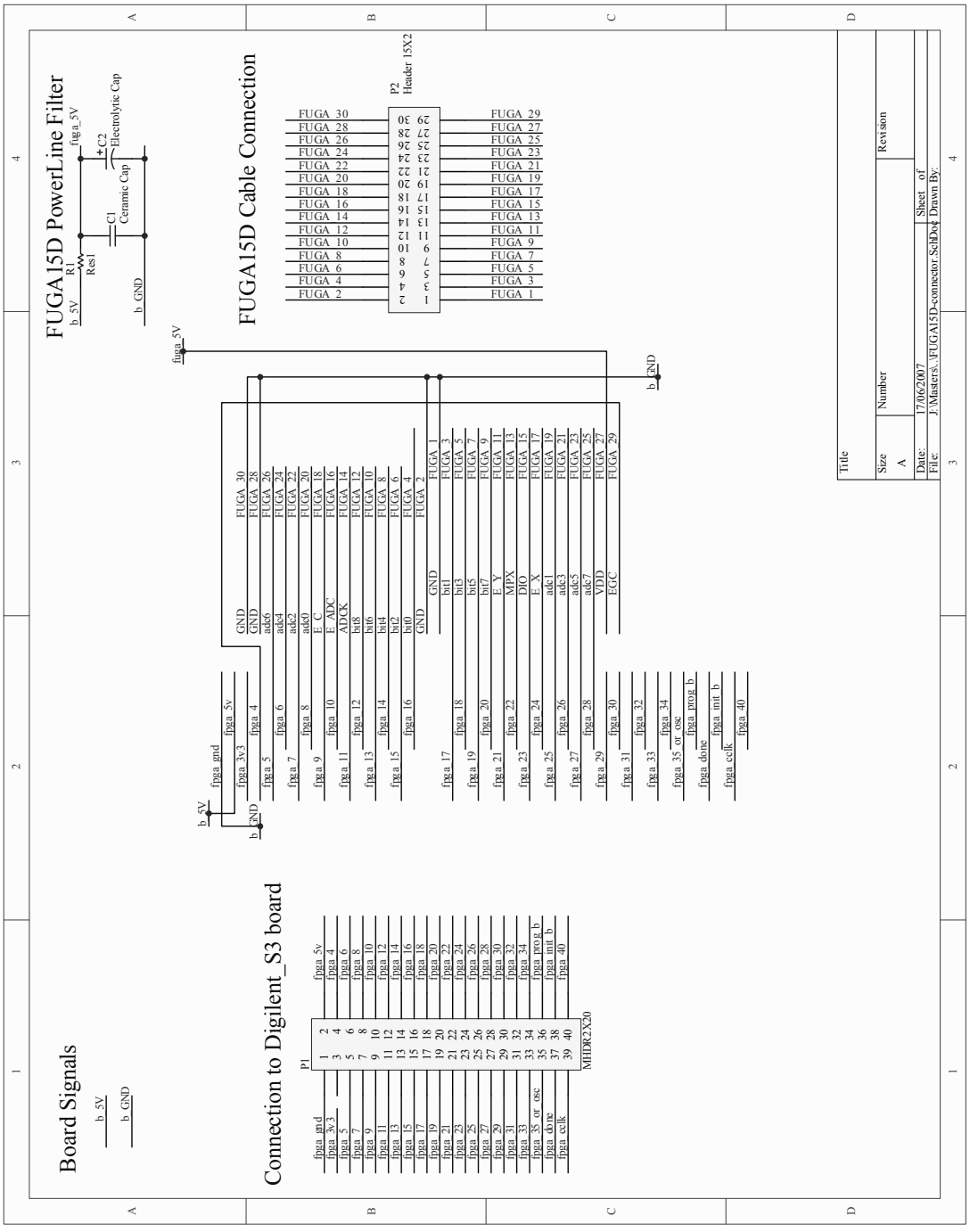
Operation	**Delays**
Cycles reading the sensor array	$(t_{clock} + t_{pix_{xy}} + t_{clock}) \times 512$ $+(t_{clock} + t_{pix_x} + t_{clock}) \times 511 \times 512$
Total	$524288 \times t_{clock} + 512 \times t_{pix_{xy}} + 261632 \times t_{pix_x}$

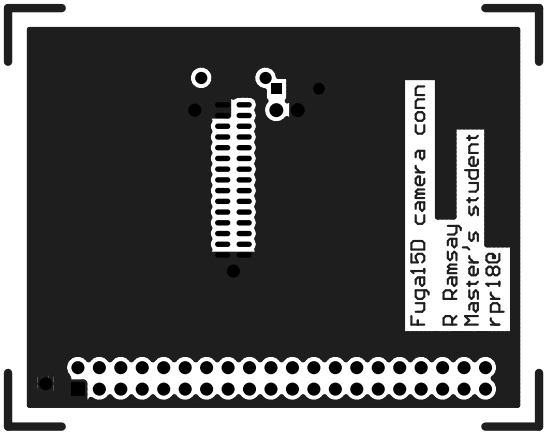
Table A.2 Read-frame delay for the CMOS Camera Controller module.

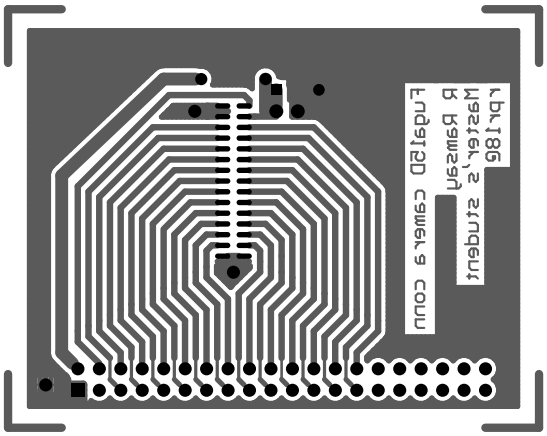
Operation	**Delays**
Cycles reading the sensor array	$(t_{clock} + t_{pix_{xy}} + t_{clock}) \times 512$ $+(t_{clock} + t_{pix_x} + t_{clock}) \times 511 \times 512$
States setting the reference row	$(2t_{clock})$ $+(t_{clock} + t_{pix_{xy}} + t_{clock})$ $+(t_{clock} + t_{fpix_x} + t_{clock}) \times 511$ $+(2t_{clock})$
Write pixel (255,255)	$(t_{clock} + t_{pix_{xy}})$
Total	$525317 \times t_{clock} + 514 \times t_{pix_{xy}} + 262143 \times t_{pix_x}$

Table A.3 Read-frame delay for the Fuga-15D specific CMOS Camera Controller module.

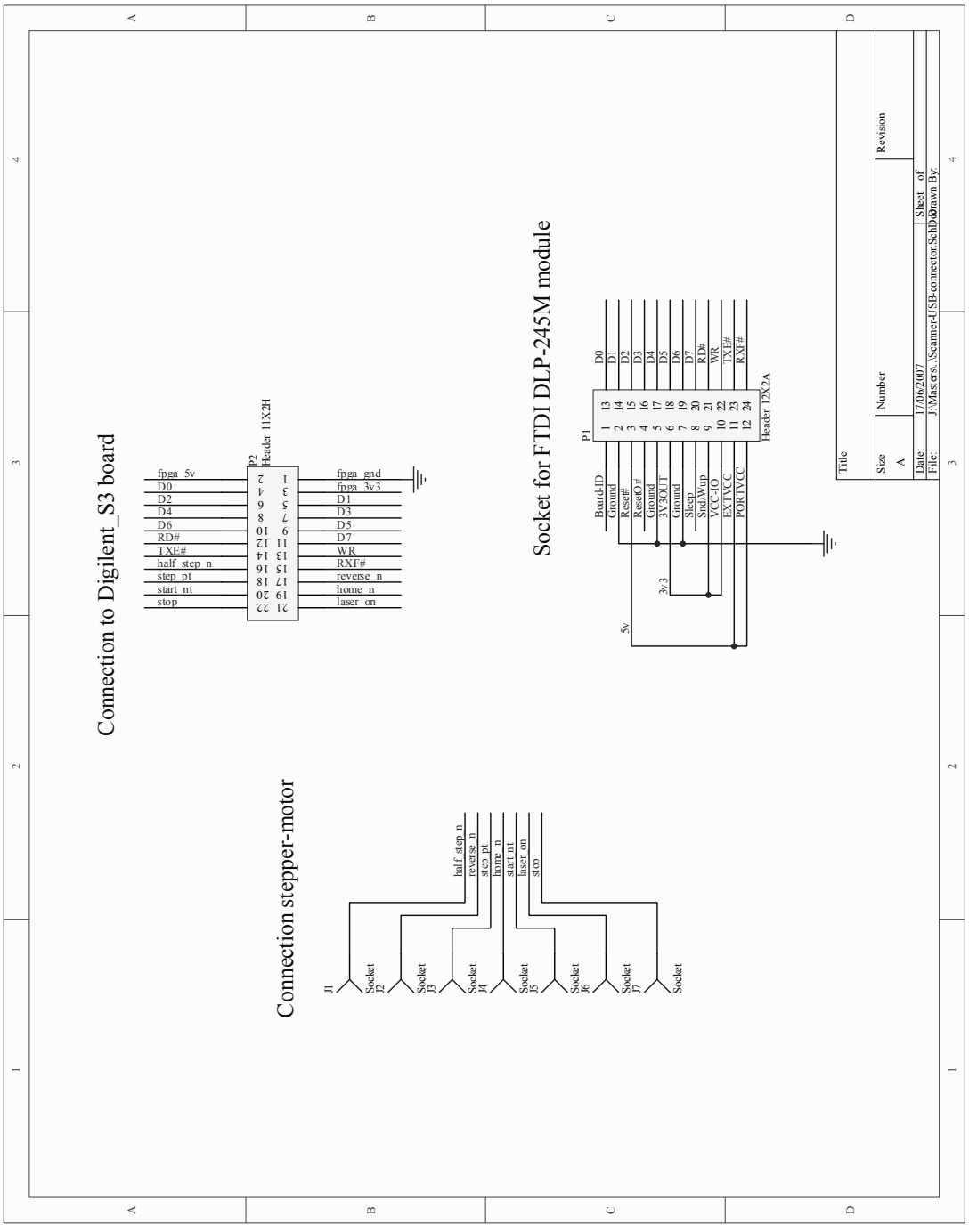
A.4 CAMERA PCB CONNECTOR

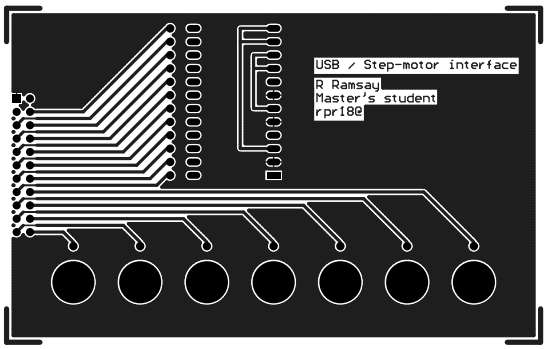


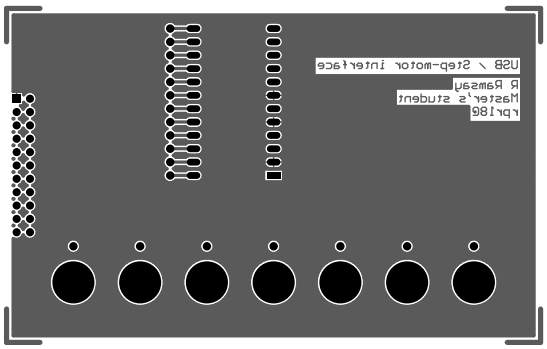

Fuga15D camera conn
R Ramsay
Master's student
rpr18@



A.5 SCANNER PLATFORM CONNECTOR




USB / Step-motor interface
R Ramsay
Master's student
rpr180



A.6 RESTORING AND NON-RESTORING DIVISION

The division algorithms outlined in [44] that were implemented in VHDL are shown below.

A.6.1 Restoring Division

Restoring division is so-called because it retains the previous partial remainder and restores it if the subtractions result is negative.

This initially loads the numerator into the partial remainder R, and the denominator into D. The algorithm then iteratively determine the quotient bits q_i using equation (9) from [44],

$$\begin{aligned} q_i &= \begin{cases} 0 & \text{if} \quad 2R_{i-1} < D \\ 1 & \text{if} \quad 2R_{i-1} \geq D \end{cases} \\ R_i &= \begin{cases} 2R_{i-1} & \text{if} \quad q_i = 0 \\ 2R_{i-1} - D & \text{if} \quad q_i = 1 \end{cases} \end{aligned} \tag{A.1}$$

Equation (10) from [44] optimises this by calculating a temporary remainder R' which performs the subtraction before q_i is known,

$$\begin{aligned} R'_i &= 2R_{i-1} - D \\ q_i &= \begin{cases} 0 & \text{if} \quad R'_i < 0 \\ 1 & \text{if} \quad R'_i \geq 0 \end{cases} \\ R_i &= \begin{cases} 2R_{i-1} & \text{if} \quad q_i = 0 \\ R'_i & \text{if} \quad q_i = 1 \end{cases} \end{aligned} \tag{A.2}$$

This then evaluates the sign of R'_i to see if the last subtraction was correct, and should be stored.

A.6.2 Non-Restoring Division

Non-restoring division uses a subtraction in the previous iteration to replace the calculation of R' used in equation A.2. If the subtraction was unnecessary it is removed in the next subtraction by adding D, since $2D - D = -D$.

These algorithms load $2N - D$ into R at the beginning of the division because the equations require a previous iteration to test. The division can be implemented using separate add and subtract units to find the remainder (equation (11) in [44])

minimising propagation delays,

$$\begin{aligned} q_i &= \begin{cases} 0 & \text{if} \quad R_{i-1} < 0 \\ 1 & \text{if} \quad R_{i-1} \geq 0 \end{cases} \\ R_i &= \begin{cases} 2R_{i-1} + D & \text{if} \quad q_i = 0 \\ 2R_{i-1} - D & \text{if} \quad q_i = 1 \end{cases} \end{aligned} \tag{A.3}$$

or it can multiplex D to reduce logic by using a single adder (equation (13) in [44]),

$$\begin{aligned} q_i &= \begin{cases} 0 & \text{if} \quad R_{i-1} < 0 \\ 1 & \text{if} \quad R_{i-1} \geq 0 \end{cases} \\ R_i &= 2R_{i-1} + \begin{cases} D & \text{if} \quad q_i = 0 \\ -D & \text{if} \quad q_i = 1 \end{cases} \end{aligned} \tag{A.4}$$

A.7 CONSTANTS USED BY THE 3D POINT CALCULATOR MODULE

fpga_pixel_width	$12.5 \times 10^{-6} \times 2^{33}$
fpga_pixel_offset_x	275.30×2^{7}
fpga_pixel_offset_y	255.50×2^{7}
point_2d_z	-0.0085954×2^{22}
fpga_laser_norm_x	0.93758×2^{17}
fpga_laser_norm_y	0×2^{17}
fpga_laser_norm_z	0.34775×2^{17}
fpga_laser_mag_r	0.27815×2^{18}
fpga_mat_11	0.99958×2^{16}
fpga_mat_12	0
fpga_mat_13	-0.02879×2^{16}
fpga_mat_14	-0.15×2^{8}
fpga_mat_21	0×2^{16}
fpga_mat_22	1×2^{16}
fpga_mat_23	0×2^{16}
fpga_mat_24	0×2^{8}
fpga_mat_31	0.028793×2^{16}
fpga_mat_32	0×2^{16}
fpga_mat_33	0.99958×2^{16}
fpga_mat_34	0.0042995×2^{8}

Table A.4 Constants used by the 3D Point Calculator Module to generate 3D data from pixels.

REFERENCES

[1] T. G. Llewellyn, "Three dimensional interior room scanner," Master's thesis, University of Canterbury New Zealand, 2001.

[2] S. Pieper, J. Rosen, and D. Zeltzer, "Interactive graphics for plastic surgery: A task-level analysis and implementation," in *Symposium on Interactive 3-D Graphics*, 1992, pp. 127–134.

[3] J.-A. Beraldin, F. Blais, M. Rioux, and L. Cournoyer, "Eye-safe digital 3-d sensing for space applications," pp. 196–211, January 2000.

[4] Google, "Google earth," 2007. [Online]. Available: http://earth.google.com/

[5] Microsoft, "Virtual earth / live maps," 2007. [Online]. Available: http://virtualearth.spaces.live.com/

[6] P. J. Besl, "Active, optical range imaging sensors," *Mach. Vision Appl.*, vol. 1, no. 2, pp. 127–152, 1988.

[7] M. W. Burke, *Image Acquisition: Hand book of Machine Vision Engineering Vol I.* New York: Chapman & Hall, 1996.

[8] R. Baribeau and M. Rioux, "Influence of speckle on laser range finders," *Applied Optics*, vol. 30, pp. 2873–2878, July 1991.

[9] Standards Association of Australia, *Laser Safety AS2211*, North Sydney, 1981.

[10] B. K. Horn, *Robot Vision.* McGraw-Hill Higher Education, 1986.

[11] E. Cilingiroglu, U.; Sicheng Chen; Cilingiroglu, "Range sensing with a scheimpflug camera and a cmos sensor/processor chip," in *Sensors Journal, IEEE, Vol.4, Iss.1*, February 2004, pp. 36–44.

[12] J. M. Sasian, "Image plane tilt in optical systems," *Optical Engineering*, vol. 31, pp. 527–532, Mar. 1992.

[13] D. Litwiller, "Cmos vs. ccd: Maturing technologies, maturing markets," in *Photonics Spectra*, August 2005.

[14] A. Theuwissen, "Building a better moustrap," in *OE Magazine*, January 2001.

[15] D. Litwiller, "Ccd vs. cmos: Facts and fiction," in *Photonics Spectra*, January 2001.

[16] J. T.McClave and T. Sincich, *Statistics*, 9th ed. Englewood Cliffs, NJ: Prentice-Hall, 2003.

[17] H. Tian, B. Fowler, and A. Gamal, "Analysis of temporal noise in cmos photodiode active pixel sensor," 2001.

[18] M. H. Izadi and K. S. Karim, "Noise analysis of a cmos active pixel sensor for tomographic mammography," in *IWSOC '05: Proceedings of the Fifth International Workshop on System-on-Chip for Real-Time Applications (IWSOC'05).* Washington, DC, USA: IEEE Computer Society, 2005, pp. 167–171.

[19] *Fuga Data Sheets*, Vector International, 1999. [Online]. Available: http://www.laseroptronix.se/sourc/vector/Fuga15.pdf

[20] S. O. Otim, D. Joseph, B. Choubey, and S. Collins, "Modelling of high dynamic range logarithmic cmos image sensors," in *Instrumentation and Measurement Technology Conference*, vol. 1, May 2004, pp. 451–456.

[21] *iMVS-157, Intelligent CMOS Programmable Camera*, AKAtech SA. [Online]. Available: http://www.akatech.ch/pdf/iMVS157_Datasheet.pdf

[22] *Basler A601f-HDR, Users Manual*, Basler Vision Technologies, March 2004. [Online]. Available: www.baslerweb.com/downloads/11723/A601f-HDR_Users_Manual_DA0

[23] *MT9V403C12ST, 1/2-INCH CMOS Active Pixel CMOS Image Sensor*, Micron Technology, Inc. [Online]. Available: download.micron.com/pdf/datasheets/imaging/MT9V403_DS.pdf

[24] *Basler A601f-HDR, Product Specifications*, Basler Vision Technologies. [Online]. Available: http://www.baslerweb.com/downloads/11684/A601602622_2_web.pdf

[25] *Basler A601f, A602f, and A622f*, Basler Vision Technologies. [Online]. Available: http://www.opsci.com/datasheets/Basler/A601f-HDR_Data_Sheet.pdf

[26] A. Schiem, private correspondance, December 2004, vC Support Europe, Basler Vision Technologies.

[27] M. Goulding, private correspondance, June 2006, total Turnkey Solutions.

[28] ——, private correspondance, August 2006, total Turnkey Solutions.

[29] *ACM100 Automotive Camera Module*, Cypress Semiconductor Corp., 2006. [Online]. Available: http://www.sensata.com/files/vision-acm100.pdf

[30] *Evaluation Camera Kit (ECK100)*, Cypress Semiconductor Corp., 2006. [Online]. Available: http://www.sensata.com/files/vision-XAECK100.pdf

[31] *CMC-1300 / C, High-Speed High-Resolution CMOS Camera*, VDS Vosskuhler GmbH. [Online]. Available: www.vdsvossk.de/download/doc/en/CMC-1300_en.pdf

[32] *MC13xx High Speed CMOS Camera*, Mikrotron GmbH, 2006. [Online]. Available: www.mikrotron.de/pdf/mc13xxman_e.pdf

[33] *M570 CameraLink Frame Grabber PMC*, Aitech Defense Systems, Inc. [Online]. Available: www.rugged.com/protected/m570.pdf

[34] *P3i_CL/PMC, Digital Data PMC Frame Grabber for Camera Link Applications*, ELTEC Elektronik AG, 2004. [Online]. Available: www.eltec.de/datasheets/p3iclpmc_0c.pdf

[35] *Spartan-3 Starter Kit Board User Guide*, Xilinx Inc., 2005. [Online]. Available: http://www.digilentinc.com/Data/Products/S3BOARD/S3BOARD-rm.pdf

[36] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," in *IEEE JOURNAL OF ROBOTICS AND AUTOMATION*, AUGUST 1987.

[37] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *CVPR97*, 1997.

[38] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.

[39] *DLP-USB245M-G USB to FIFO Parallel Interface Module*, DLP Design, 2002. [Online]. Available: www.ftdichip.com/Documents/DataSheets/ds245b16.pdf

[40] *FT245BM USB FIFO (USB - Parallel) I.C.*, Future Technology Devices Intl. Limited, 2005. [Online]. Available: www.ftdichip.com/Documents/DataSheets/DS_FT245BM.pdf

[41] *XST User Guide*, Xilinx Inc. [Online]. Available: http://toolbox.xilinx.com/docsan/xilinx9/books/docs/xst/xst.pdf

[42] *Using Embedded Multipliers in Spartan-3 FPGAs*, Xilinx Inc., 2003. [Online]. Available: www.xilinx.com/bvdocs/appnotes/xapp467.pdf

[43] *Spartan-3 FPGA Family: Complete Data Sheet*, Xilinx Inc., 2007. [Online]. Available: direct.xilinx.com/bvdocs/publications/ds099.pdf

[44] D. G. Bailey, "Space efficient division on fpgas," in *Electronics New Zealand Conference (EnzCon'06), Christchurch, NZ*, November 2006, pp. 206–211.

[45] "Sensata technologies inc." [Online]. Available: http://www.sensata.com